

Diplomarbeit

Interaktive 3D-Visualisierung eines Projekts des Fraunhofer-Instituts für Produktionstechnik und Automatisierung

vorgelegt von

Robert G. Rossi

Matrikel-Nr. 10181

an der
Fachhochschule Stuttgart
Hochschule der Medien
Studiengang Audiovisuelle Medien

1. Prüfer: Prof. Dr. Thomas Keppler
Fachhochschule Stuttgart
Hochschule der Medien

2. Prüfer: Prof. Uwe Schulz
Fachhochschule Stuttgart
Hochschule der Medien

Meinen Dank für die Betreuung am Fraunhofer Institut für Produktionstechnik und Automatisierung richte ich an

Dipl.-Ing. (FH) Andreas Hähnke

Dipl.-Ing. Markus Mersinger

Hiermit versichere ich, die vorliegende Diplomarbeit selbständig und ohne fremde Hilfe verfasst zu haben. Sämtliche Quellen sind im Text oder im Anhang nachgewiesen.

Robert G. Rossi

Inhaltsverzeichnis

Danksagung und Eidesstattliche Erklärung	II
Dieses Inhaltsverzeichnis.....	III
Abbildungsverzeichnis	V
Anlagenverzeichnis	VII
Tabellenverzeichnis	VII
Abkürzungsverzeichnis.....	VI
1. Einleitung.....	1
1.1 Die Motivation	1
1.2 Inhalt und Ziele der Arbeit	2
1.2.1 Inhalt	2
1.2.2 Ziel: Erfahrungsbericht	3
1.2.3 Ziel: Dokumentation	3
1.3 Vorstellung des Instituts für Produktionstechnik und Automatisierung	3
1.4 Begriffliche Einführung	4
1.5 Zielspezifikation.....	5
1.5.1 Analyse der Ausgangssituation.....	5
1.5.2 Ableitung der Anforderungen	10
2. Die Produktion	15
2.1 Auswahl und Bewertung von 3D-Autorenwerkzeugen.....	15
2.1.1. Kriterien für die Auswahl	15
2.1.2 Eyematic <i>Shout3D 2 Beta 4</i> und Virtock <i>Spazz3D Version 2.4 c</i>	19
2.1.3 Parallel Graphics: <i>Internet Scene Assembler (ISA) v1.0</i>	22
2.1.4 Cryo Networks: <i>Site Construction Set (SCS)</i>	24
2.1.5 Nemo: <i>NeMo Creation 1.0.1</i>	25
2.1.6 Übersicht der untersuchten Produkte	28
2.2 Eine Schritt-für-Schritt-Anleitung	29

Inhaltsverzeichnis

2.2.1 Die 3D-Modelle - von Medit zu Creation	29
2.2.2 Die Texturen und Materialien	36
2.2.3 Die Animationen	43
2.2.4 Die Interaktivität	48
2.2.5 Die Webseite als Benutzeroberfläche	54
2.3 Weitere verwendete Softwares	57
3. Schlussbetrachtung	59
4. Literatur- und Quellenverzeichnis	61
5. Anlagen	62

Abbildungsverzeichnis

Abbildung 1: Screenshot der 3D-Ansicht der SVP und Foto der echten Anlage	6
Abbildung 2: Architektur der Simulationsplattform.....	7
Abbildung 3: Systemarchitektur der Simulationsplattform	8
Abbildung 4: Ortsfeste Sicht für Diagrammauswahl.....	9
Abbildung 5: Benutzeroberfläche der interaktiven 3D-Visualisierung.....	13
Abbildung 6: Benutzeroberfläche des <i>Shout3DWizard</i>	20
Abbildung 7: Benutzeroberfläche von <i>Spazz3D</i>	21
Abbildung 8: Benutzeroberfläche des <i>Internet Scene Assembler</i>	22
Abbildung 9: Benutzeroberfläche von <i>Site Construction Set</i>	24
Abbildung 10: Benutzeroberfläche von <i>NeMo Creation 1.0.1</i>	26
Abbildung 11: „Branding“ der „Educational Version“ von <i>NeMo Creation</i>	28
Abbildung 12: <i>Polygon Reduction Editor</i> von <i>CosmoWorlds</i>	32
Abbildung 13: Bildfehler durch doppelte Polygone in der perspektivischen Ansicht und im gerenderten Bild	33
Abbildung 14: Selektionswerkzeug von <i>3D Studio Max</i> : Auswahl von <i>Dummy</i> -Objekten	35
Abbildung 15: Textur-Setup und Alpha-Maske für das Geländer in <i>NeMo</i>	37
Abbildung 16: Material-Setup in <i>NeMo</i>	37
Abbildung 17: Geländertextur in <i>NeMo</i> JPG-komprimiert / unkomprimiert	40
Abbildung 18: Zusammenfügen von Modellen und ihren Materialien in <i>3D Studio Max</i>	41
Abbildung 19: Material-Liste des Casters nach dem VRML-Import und nach dem Zusammenfassen von Materialien und Modellen	42
Abbildung 20: <i>Building Block</i> in <i>NeMo</i>	45
Abbildung 21: <i>NeMo-Building Block</i> „Bramme erzeugen“ zur Skalierung einer Bramme auf die gewünschte Länge	46
Abbildung 22: <i>NeMo</i> -Skript „Time Manager“ zur Einstellung des Zeitfaktors.....	51
Abbildung 23: Benutzerhinführung zur Interaktiven 3D-Visualisierung.....	55

Anlagenverzeichnis

Anlage A: Alphabetisches Verzeichnis der NeMo Building Blocks	62
Anlage B: Screenshot des Outline-Editors von CosmoWorlds	70
Anlage C: Tabelle THREED-Events der Integrierten Simulations- und Visualisierungs-Plattform.....	71
Anlage D: NeMo-Skript „Event Manager“	73
Anlage E: NeMo-Skript „ferry manager“	74
Anlage F: CD-ROM	

Tabellenverzeichnis

Tabelle 1: Übersicht der untersuchten Produkte hinsichtlich der Anforderungen an die Interaktive 3D-Visualisierung.....	29
Tabelle 2: Ereignisse in der Interaktiven 3D-Visualisierung.....	48

Abkürzungsverzeichnis

AVI	=	Audio Visual Interleave
BMP	=	Bitmap
CD-ROM	=	Compact Disk-Read Only Memory
DIB	=	Device Independent Bitmap
DLL	=	Dynamic Link Library
GUI	=	Graphical User Interface
HTML	=	Hyper Text Markup Language
IPA	=	Institut für Produktionstechnik und Automatisierung
ISA	=	Internet Scene Assembler
IT	=	Informationstechnik
JPG	=	Joint Photographic Expert Group
kB	=	Kilobyte
LOD	=	Level of Detail
MIDI	=	Music Instrument Digital Interface
OS	=	Operating System
PC	=	Personal Computer
PCX	=	PiCture eXchange
PDF	=	Portable Document Format
PDM	=	Produktdaten-Management
PHP	=	PHP Hypertext Preprocessor
PR	=	Public Relation
RDBMS	=	Relationales Datenbank-Managementsystem
RGB	=	Red Green Blue bzw. Rot Grün Blau
SCOL	=	Standard Cryo Online Language
SCS	=	Site Construction Set
SGI	=	Silicon Graphics Inc.
SMS	=	Schloemann-Siemag
SQL	=	Structured Query Language
SVP	=	Integrierte Simulations- und Visualisierungs-Plattform
TGA	=	Targa
TIFF	=	Tagged Image File Format
URL	=	Unified Resource Locator
UV	=	Koordinaten für das Skalieren, Verschieben oder Kacheln von Texturen auf 3D-Modellen
VRML	=	Virtual Reality Modeling Language
W3C	=	World Wide Web Consortium
WWW	=	World Wide Web
X3D	=	Extensible 3D

1 Einleitung

1.1 Die Motivation

Bei der vorliegenden Arbeit handelt es sich um die Produktion einer interaktiven 3D-Visualisierung zu Präsentationszwecken für das Fraunhofer Institut für Produktionstechnik und Automatisierung (IPA). Mit der interaktiven 3D-Visualisierung wird die vom IPA entwickelte „Integrierte Simulations- und Visualisierungplattform“ (SVP) vorgestellt. Für die Motivation dieser Arbeit stand zunächst nicht im Vordergrund, *was* interaktiv und in 3D visualisiert wird, sondern *wie* und *womit*. Das für dieses Projekt eingesetzte 3D-Autorenwerkzeug *NeMo* ist ein Produkt des französischen Unternehmens Virtools.

Zwei Motive waren für diese Arbeit ausschlaggebend:

Das erste Motiv entspringt der Beobachtung, dass die Selbstdarstellung einer Organisation im WWW mittels Bild und Text längst nicht mehr ausreichend ist. Weitere Medienformate sind inzwischen durch große Marktpräsenz oder durch überzeugende Qualität zu Standards gewachsen und haben auf Webseiten Verbreitung gefunden. Als Beispiele sind *RealMedia* für Video und Audio, *Macromedia Flash* für interaktive 2D-Animationen oder das vom Fraunhofer Institut für Integrierte Schaltungen entwickelte Audio-Format *mp3* zu nennen.

Das zweite Motiv ist subjektiver Art und entstammt der persönlichen Neigung und dem Interesse des Autors für die Kombination von Computeranimationen und interaktiven (Online-)Anwendungen.

Mit dem Web-Auftritt versucht eine Organisation u. a. dem Image oder der Branche entsprechend die eigene Position im Markt zu unterstreichen. Für manche Marktteilnehmer sind interaktive 3D-Animationen das adäquate Medium, um dem Besucher über den Web-Auftritt Produkte oder Aktivitäten nahe zu bringen.

Durch einen Kontakt bei der Pressestelle des Fraunhofer Instituts für Produktionstechnik und Automatisierung (IPA) ergab sich für den Autor die Möglichkeit, das Medium „Interaktive 3D-Animationen“ kennen zu lernen und damit produktiv zu arbeiten.

1.2 Inhalt und Ziele der Arbeit

1.2.1 Inhalt

Nach der Vorstellung des IPA in Abschnitt 1.3 und einer begrifflichen Einführung in Abschnitt 1.4 werden in Abschnitt 1.5 die Ausgangssituation analysiert, Ziel-spezifikationen für die Interaktive 3D-Visualisierung ermittelt und daraus Kriterien zur Auswahl eines 3D-Autorenwerkzeugs abgeleitet.


Das Kapitel 2 führt eine Reihe von 3D-Autorenwerkzeugen auf, die auf ihre Verwendungs-möglichkeiten für die Produktion der Interaktiven 3D-Visualisierung geprüft werden.

Anschließend folgt eine Dokumentation der Arbeitsschritte auf dem Weg von den solitären 3D-Modellen über die zentralen Animationen bis hin zur Integration der fertigen 3D-Szene in die Webseite des IPA.

Abgeschlossen wird Kapitel 2 mit einer kurzen Vorstellung der weiteren im Projekt verwendeten Softwares.

Das Kapitel 3 schließt die Arbeit mit einem Fazit ab.

Große Abbildungen und Tabellen, die den Textfluss stören würden, sind in den Anlagen enthalten.

Absätze, die mit einer Glühbirne  gekennzeichnet sind, enthalten Erkenntnisse, die während der praktischen Arbeit gewonnen wurden.

Der praktische Teil der Arbeit hatte folgende Ziele:

- Die Produktion der Interaktiven 3D-Visualisierung und damit einhergehend
- das Erforschen der Möglichkeiten und Grenzen bei der Zusammenführung der Medien „Interaktiv/Online“ und „3D“ anhand eines ausgewählten Autoren-Werkzeugs.

In erster Linie wird die Interaktive 3D-Visualisierung auf der Webseite der IT-Gruppe der Abteilung „Technische Produktionsplanung“ beim IPA dazu verwendet werden, dem Besucher einen interessanten und spielerischen Einblick in die Projektarbeit zu geben.

1.2.2 Ziel 1: Erfahrungsbericht:

Der Erfahrungsbericht wendet sich an Leser, die generell an dem Medium „Interaktive 3D-Animationen“ interessiert sind. Von besonderem Interesse für diese Leser werden folgende Kapitel und Abschnitte sein:

- 1.3 Vorstellung des Instituts für Produktionstechnik und Automatisierung
- 1.4 Begriffliche Einführung
- 1.5 Zielspezifikation
- 2.1 Auswahl und Bewertung von 3D-Autorenwerkzeugen
- 2.3 Verwendete Softwares
- 3. Schlussbetrachtung.

Die technische Kenntnis der Produktion von 3D-Animationen bzw. Programmierkenntnisse sind nicht Voraussetzung für das Verständnis der o. g. Kapitel und Abschnitte.

1.2.3 Ziel 2: Dokumentation

Die Dokumentation richtet sich an die Mitarbeiter des IPA, die künftig mit der Weiterentwicklung der bestehenden Interaktiven 3D-Visualisierung oder mit weiterführenden Projekten betraut sein werden. Insbesondere Abschnitt 2.2 nimmt sich der Details an. Hier wird vorausgesetzt, dass der Leser Kenntnisse in der Bildbearbeitung, in der 3D-Modellierung, insbesondere mit *3D Studio Max*, und Programmierkenntnisse in JavaScript, PHP und HTML besitzt. Darüber hinaus ist das Verständnis von ereignis- und objektorientierter Programmierung von großem Vorteil, da das für die Realisierung eingesetzte 3D-Autoren-Werkzeug *NeMo* auf diesen Konzepten basiert.

1.3 Vorstellung des Instituts für Produktionstechnik und Automatisierung

Das Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA wurde 1959 in Stuttgart gegründet und beschäftigt heute rund 200 Mitarbeiterinnen und Mitarbeiter.

Einleitung

Begriffliche Einführung

Forschungs- und Entwicklungsschwerpunkte des IPA sind organisatorische Aufgabenstellungen aus dem Produktionsbereich von Industrieunternehmen. Zunehmend entstehen Aufgabengebiete aus den Entwicklungen und Veränderungen im Dienstleistungssektor.

Durch die Forschungs- und Entwicklungsarbeiten werden in den Unternehmen Automatisierungs- und Rationalisierungspotenziale aufgezeigt, die zu kostengünstigen und umweltfreundlichen Produktionsabläufen sowie verbesserten Produkten führen und so die internationale Wettbewerbsfähigkeit und die Arbeitsplatzsituation verbessern.

Dazu werden in den Bereichen

- Unternehmensmanagement
- Automatisierung und
- Produktionstechniken

des IPA Methoden, Komponenten und Geräte bis hin zu kompletten Maschinen und Anlagen entwickelt, erprobt und exemplarisch eingesetzt.

Auftraggeber sind überwiegend Privatunternehmen. Daneben werden Projekte bearbeitet, die im Rahmen öffentlicher Forschungsprogramme gefördert werden.¹

Die vorliegende Diplomarbeit wurde in der Abteilung „Technische Produktionsplanung“ des Bereichs Automatisierung erstellt, deren Leistungen u. a. die Durchführung von Simulationen, den Aufbau und die Gestaltung virtueller Produktionsanlagen oder Fabriken aus Datenbanken und Produktdaten-Management-Systemen umfasst.

1.4 Begriffliche Einführung

Die Ausführungen in dieser Arbeit bedienen sich der Ausdrücke *3D-Autorenwerkzeuge* und *3D-Autor*. Analog zu eingeführten Software-Produkten wie beispielsweise Macromedia *Director* oder *Flash*, die beide Werkzeuge zur Produktion von interaktiven 2D-Inhalten darstellen, sind 3D-Autorenwerkzeuge Programme, mit denen interaktive 3D-Inhalte produziert werden. Somit sind sie

¹ Vgl. www.ipa.fhg.de

abzugrenzen von 3D-Programmen wie beispielsweise Alias|Wavefront *Maya* oder Kinetix *3D Studio Max* für die Produktion von 3D-Filmen.

Wenn es um 3D-Modelle im allgemeinen geht, wird der Begriff *Modell* verwendet. Im Abschnitt 2.2.1, der sich speziell mit der Bearbeitung von 3D-Modellen beschäftigt, wird hingegen genauer unterschieden zwischen einem *Modell* als logisches Objekt und der formgebenden *Geometrie* dieses Objekts.

Desweiteren sind Eigennamen von Produkten immer *kursiv* gedruckt. Die Begriffswelt der Produkte wird übernommen, um die Ausführungen in der Praxis leichter nachvollziehen zu können. Solche Begriffe sind ebenfalls *kursiv* gedruckt.

Während diese Arbeit entstand, wurde vom Web3D Consortium² im August 2001 ein offener Standard namens X3D verabschiedet. X3D ist der Nachfolger des Standards VRML97 zur Beschreibung virtueller Welten für das WWW.

X3D wird allerdings erst noch mit Leben zu füllen sein. Führende Software-Hersteller wie blaxxun werden bald erste Produkte dafür anbieten. Für VRML97 hingegen gibt es bereits eine Reihe von Produkten, von denen einige im Abschnitt 2.1 vorgestellt werden.

Wie der Name des Web3D Consortiums andeutet, ist Web3D ein eher allgemeiner Begriff für den Einsatz von 3D-Bildern im WWW.

1.5 Zielspezifikation

1.5.1 Analyse der Ausgangssituation

Die SVP wurde vom IPA am Beispiel einer zweistrangigen Gieß-Walz-Anlage der SMS Demag AG³ realisiert. Die SMS Demag AG ist ein weltweit tätiger Hersteller von Hütten- und Walzwerkanlagen. Die Gieß-Walz-Anlage produziert sog. Dünnbrammen. Das sind 45-70 mm dicke und bis zu 50 m lange Stahlteile, die zur Fertigung z. B. von Feder-, Kugellager-, Ventil- und anderen Spezialstählen verwendet werden. Die Oberflächen- und Innenqualität der Dünnbrammen muss

² Das Web3D Consortium arbeitet eng zusammen mit dem World Wide Web Consortium (W3C).

³ Vgl. www.sms-demag.de

Einleitung

Zielspezifikation

insbesondere hohen risskritischen Stahlgüten entsprechen. Die SMS Demag AG stellt Gieß-Walz-Anlagen als Ein- oder Zweistranganlagen her.⁴

Im Dünnbrammen-Produktionsprozess auf einer Zweistranganlage wird der zunächst flüssige Stahl aus zwei Drehtürmen gegossen, auf die gewünschte Länge geschnitten und in Ofenstränge befördert. Dort werden die Stahlteile auf die für das Walzen erforderlichen Temperaturen erhitzt und auf Rollstrecken mit einer Geschwindigkeit von bis zu 5 m/min befördert. Anschließend werden die Stahlteile beider Ofenstränge über Schwenkfähren dem Walzwerk zugeführt, wo sie mit hoher Geschwindigkeit durch mehrere (im Beispiel sechs) Walzmühlen befördert und darin auf die gewünschte Dicke gewalzt werden. Die darauf folgende Kühlstrecke kühlt die Stahlteile ab und gibt sie zur weiteren Verarbeitung ab. Dieser Ablauf ist in der SVP abgebildet.



Bild 1: Links: Screenshot der 3D-Ansicht der SVP, rechts: Foto der echten Anlage. Zu sehen ist das Walzwerk.

Die SVP wurde entwickelt, um zusätzliche Dienstleistungen zu ermöglichen:

- visuelles Demonstrationswerkzeug im Vertrieb
- Schulungswerkzeug bei der Inbetriebnahme von Anlagen
- während der Nutzung von Zweistranganlagen, um Betriebsstrategien zu optimieren

⁴ Eine Zweistranganlage ermöglicht eine höhere Produktionsflexibilität und Produktivität bei einem Investitionsaufwand, der wesentlich geringer ist als der Aufwand für zwei Einstranganlagen, da bei einer Zweistranganlage **ein** Walzwerk ausreicht.

Einleitung

Zielspezifikation

Ein wesentlicher Bestandteil der Betriebsstrategieoptimierung ist die Simulation von Auftragsreihenfolgen, d. h. die Reihenfolge, in der die Stahlteile das Walzwerk durchlaufen.

Die SVP ist ein Netzwerk von mehreren handelsüblichen, aber leistungsstarken PCs. Das folgende Bild stellt die Systemarchitektur und –komponenten dar:

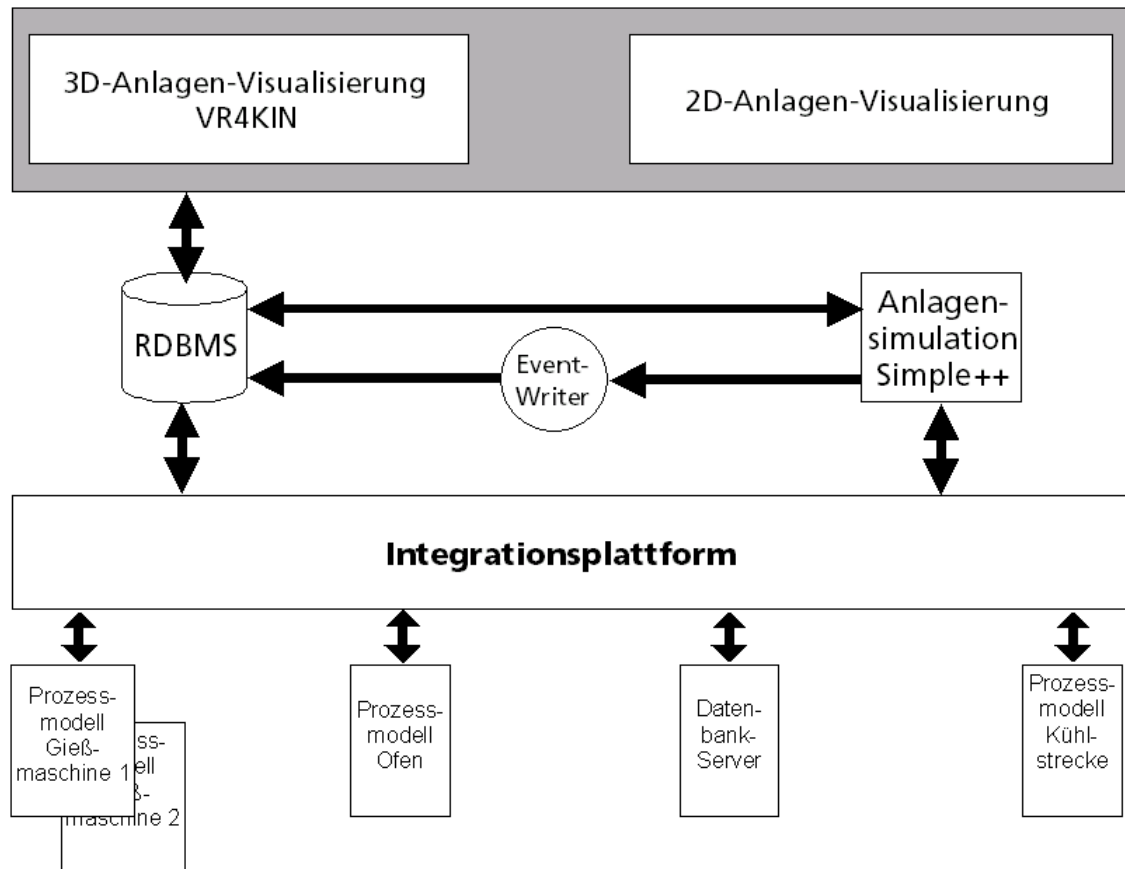


Bild 2: Architektur der Simulationsplattform (Bildquelle: Dokumentation Simulationsplattform, IPA)

In diesem Gesamtsystem interagieren neben den Prozeßmodellen folgende Komponenten:

- Integrationsplattform
- Relationale Datenbank
- Anlagensimulation
- Eventwriter
- 3D-Anlagenvisualisierung
- 2D-Anlagenvisualisierung

Einleitung

Zielspezifikation

Die aufgelisteten Komponenten stellen ausführbare Programme dar. Die Komponenten verteilen sich auf das Netzwerk mit der folgenden Konfiguration:

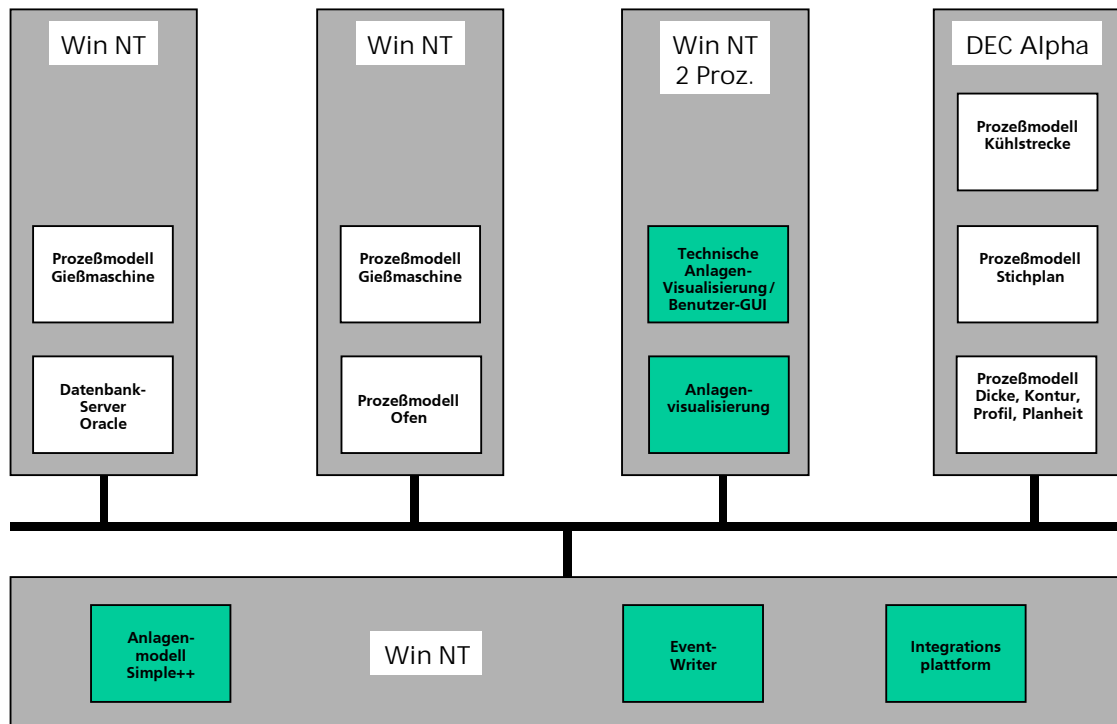


Bild 3: Systemarchitektur der Simulationsplattform (Bildquelle: Dokumentation Simulationsplattform, IPA)

Die SVP verwendet einen dedizierten Rechner mit 2 Hauptprozessoren für die Visualisierung. Damit steht ihr die komplette Rechenleistung eines Hauptprozessors und einer OpenGL-Grafikkarte⁵ zur Verfügung.

Der Anwender hat vielfache Möglichkeiten, die Simulation und die Visualisierung zu konfigurieren und sowohl vor dem Start als auch zur Laufzeit zu beeinflussen. Beispielsweise beinhaltet die Konfiguration die Einstellung verschiedener Diagnose-Stufen der Integrationsplattform; zur Laufzeit kann der Benutzer bestimmen, welche Diagrammart in verschiedenen Sichten angezeigt werden sollen. Es gibt folgende Sichten:

⁵ OpenGL ist ein plattformübergreifender Standard zur schnellen Berechnung von 3D-Bildern durch Hardware. OpenGL enthält einen Satz von Geometrieprimitiven (Punkte, Linien, Polygone) und Texturen sowie Programmierungsbefehle zu ihrer Spezifizierung in zwei oder drei Dimensionen und zur Berechnung der Bilder (vgl. OpenGL Survival Kit Tutorial, Nicole Deflaux Terry, www.eecs.tulane.edu/www/Terry/OpenGL/Introduction.html).

Einleitung

Zielspezifikation

- Prozessparametersicht:
Anzeige und Änderbarkeit von Parametern einzelner Anlagenteile
- Ortsfeste Sicht:
Zeigt Daten für Brammen an einem ausgewählten Anlagenpunkt an
- Operatorsicht:
Zeigt Daten für die zuletzt an einem Anlagenpunkt vorbei gekommene Bramme an.
- Brammenverfolgungssicht:
Zeigt Daten für eine ausgewählte Bramme an.

Die Sichten sind während der Simulationslaufzeit umschaltbar. Beim Umschalten ändert sich die 2D-Ansicht der Anlage. Im folgenden Bild ist z. B. die ortsfeste Sicht eingestellt.

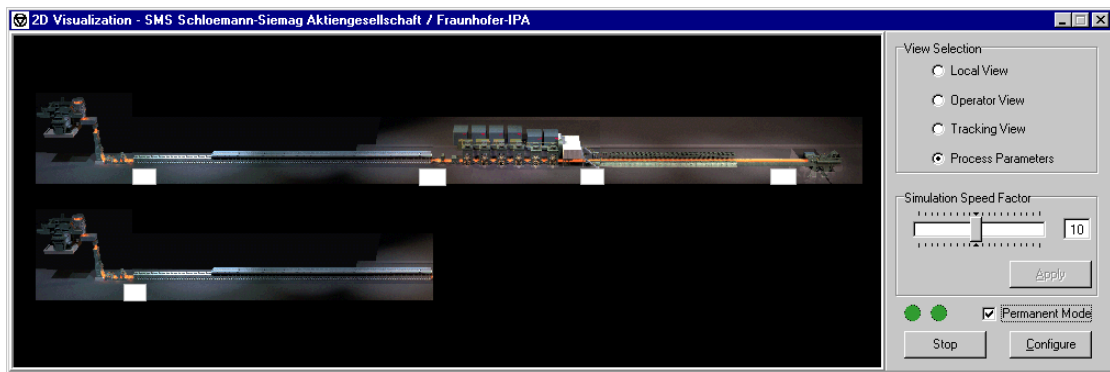


Bild 4: Ortsfeste Sicht für Diagrammauswahl⁶

Insbesondere bei der ortsfesten Sicht hat der Anwender die Möglichkeit, sich z. B. die Brammendicke oder –geschwindigkeit für jede Bramme an einem bestimmten Ort anzeigen zu lassen. In der Brammenverfolgungssicht sind Brammendicke, –geschwindigkeit und –temperatur einer ausgewählten Bramme anzeigbar. Die Bramme lässt sich aus einer Liste oder direkt in der 3D-Visualisierung auswählen.

In der 3D-Visualisierung selbst hat der Benutzer die Möglichkeit, mit einer feststehenden virtuellen Kamera zu zoomen. Der Zoom kann per Auswahl mittels Mausklick auch auf Anlagenteile oder Brammen eingestellt werden. So ist eine nähere Betrachtung der Anlagenteile möglich, allerdings nur aus der Position der feststehenden Kamera.

⁶ Diese schematische Ansicht der Anlage wird als authentisches Bedienelement in der Interaktiven 3D-Visualisierung verwendet.

Darüber hinaus ist die Simulationsgeschwindigkeit über die Zeitrafferfaktoren 1, 10 und 100 einstellbar.

1.5.2 Ableitung der Anforderungen

Die Interaktive 3D-Visualisierung auf der Webseite des IPAs soll weder etwas simulieren noch eine realitätsnahe Darstellung anbieten. Das wäre ein unangemessener Aufwand für reine PR-Zwecke. Die Interaktive 3D-Visualisierung ist ein stark vereinfachtes Abbild der SVP und soll zeigen, was die SVP beinhaltet und ein ähnliches „Look and Feel“ der Visualisierung vermitteln. Zudem soll sie in zweierlei Hinsicht so flexibel wie möglich sein:

- In gewissen Grenzen soll die Visualisierung variabel sein und sozusagen die „Simulation simulieren“. Konkret bedeutet das, dass zu Beginn die Anzahl der Dünnbrammen, ihre Längen, sowie die Auftragsreihenfolge vom Benutzer bestimmt werden können. Der Benutzer übernimmt also auf spielerische Art und Weise - durch Probieren - die Aufgabe der SVP, eine optimale Auftragsreihenfolge zu finden. Diese Variabilität im Animationsablauf setzt eine gewisse „Intelligenz“ der 3D-Szene voraus, da die Auftragsreihenfolge bestimmt, wann welche Dünnbramme über die Schwenkfähren dem Walzwerk zugeführt wird.
- Die Interaktive 3D-Visualisierung soll nach Möglichkeit nicht nur auf der Webseite des IPA, sondern auch für Offline-Präsentationen, z. B. bei Kunden, eingesetzt werden. Das bedeutet, dass sie in einem Standard-Browser auf einem durchschnittlichen PC gut funktionieren muss. Voraussetzung dafür ist, dass das eingesetzte 3D-Autorenwerkzeug die Möglichkeit unterstützt, die 3D-Szene offline abspielen zu können. Dies ist bei *NeMo* der Fall.

Für die Interaktive 3D-Visualisierung ist davon auszugehen, dass der Besucher der IPA-Webseite mit einem durchschnittlichen Büro-PC, i. d. R. *ohne* OpenGL-Grafikkarte, ausgestattet ist. Die Prozessor-Leistung wird vermutlich zwischen 400 und 1000 MHz betragen. Der PC ist – im Gegensatz zur SVP, die aus einem Rechnerverbund besteht - alleine für das Abspielen der Interaktiven 3D-Visualisierung zuständig. Möglicherweise sind auf diesem PC gleichzeitig andere Anwendungen aktiv, die Systemressourcen in Anspruch nehmen. Das hat zwei wichtige Folgen:

- Die 3D-Modelle müssen möglichst polygonarm sein.⁷
- Der Rechenaufwand für die Darstellung des 3D-Bildes steigt mit der absoluten Größe des sog. Viewports, also des Bild-Rahmens, gemessen in Bildschirm-Bildpunkten. Beispiel: Ein Bild mit z. B. 480 x 360 Bildpunkten benötigt bei gleicher Polygonzahl etwa die vierfache Rechenzeit des gleichen Bildes mit 240 x 180 Bildpunkten, da das erstgenannte Bild vier mal so viele Bildpunkte wie das zweite Bild enthält. Die Größe des Viewports stellt also einen Kompromiss dar zwischen Detailtreue/Detailreichtum in der Darstellung und Rechenaufwand.

Die Anlagenteile liegen bereits als 3D-Modelle vor. Die Gesamtpolygonzahl beträgt rund 350.000. Bei dem Versuch, alle Modelle sukzessive in eine 3D-Software zur Weiterverarbeitung zu laden, stellte sich schnell heraus, dass diese Zahl viel zu hoch ist, da ein Arbeiten ab ca. 200.000 Polygonen nicht mehr möglich war. Dies war ein deutlicher Hinweis, dass für die Interaktive 3D-Visualisierung noch weit weniger Polygone zulässig sein würden. Eine extreme Reduzierung der Polygonzahl ist die Folge.

Somit muss ein Verlust an Detailtiefe akzeptiert werden, allerdings in einem vertretbaren Umfang. Anders als bei der SVP wird fast vollständig auf die Verwendung von Levels of Detail⁸ (LOD) verzichtet, wodurch bereits der größte Teil an Polygonen „eingespart“ werden kann. Der erste und detaillierteste LOD mancher Objekte enthält bis zu zehn mal so viele Polygone wie der dritte oder, wenn vorhanden, vierte LOD. Um dieses drastische Einsparungspotenzial zu nutzen, werden fast nur die am wenigsten detaillierten LODs der Objekte für die Interaktive 3D-Visualisierung verwendet. Um die LOD-Funktion einmal auszuprobieren, werden lediglich die Treppen des Messhauses mit zwei LODs dargestellt.

Was die Benutzeroberfläche der Interaktiven 3D-Visualisierung angeht, so ist es prinzipiell möglich, dem Benutzer annähernd die gleichen Anzeige- und Bedien-

⁷ Die Polygonzahl ist nicht das einzige Kriterium. Ein weiteres Kriterium ist die Größe eines Polygons, gemessen in Bildschirmpunkten. Mit steigender Größe eines Polygons erhöht sich auch der Rechenaufwand für dieses Polygon. Eine diesbezügliche Optimierung der Anlagenteilmodelle hätte jedoch den zeitlichen Rahmen dieser Arbeit gesprengt.

⁸ Die Verwendung von LODs ist ein Mittel, um den Rechenaufwand für 3D-Geometrien in Grenzen zu halten und dabei möglichst wenig Darstellungsqualität, d. h. Detailreichtum, einzubüßen. Dabei werden Objekte, die sich in einer bestimmten Entfernung von der virtuellen Kamera befinden, durch Objekte mit weniger Details, d. h. weniger Polygonen, ersetzt. Die aufwändige Berechnung von Polygonen, die evtl. ohnehin nur durch ein oder sehr wenige Pixel dargestellt würden, entfällt somit. Wird zur Erzielung einer besseren Darstellungsqualität außerdem Anti-Aliasing (=Kantenglättungsverfahren) eingesetzt, vermindert sich auch der Rechenaufwand dafür erheblich.

elemente anzubieten wie in der SVP. Jedoch ergeben sich folgende Einschränkungen:

- Programmieraufwand im Rahmen dieser Arbeit, z. B. für die Umstellung des Zeitfaktors zur Laufzeit
- Fehlende „echte“, d. h. simulierte, Prozessdaten. Das bedeutet, dass z. B. in der Prozessdatendiagrammanzeige nur exemplarische Diagramme zu sehen sein werden.
- Einschränkungen beim Datenaustausch zwischen der interaktiven 3D-Visualisierung und der HTML-Seite, in die sie eingebettet ist. Diese werden im Abschnitt 2.2.4 eingehend diskutiert.

In der SVP sind nicht alle Funktionen unter einer einheitlichen Benutzeroberfläche zusammen gefasst. So werden z. B. Zahl und Länge der Dünnbrammen „von Hand“ in eine Datenbank-Tabelle eingetragen; gestartet wird die SVP durch den Aufruf mehrerer Programme, darunter die 3D-Visualisierung. Das bedeutet, dass auch die Benutzeroberfläche der Interaktiven 3D-Visualisierung stark vereinfacht und fachfremden Personen zugänglich gemacht werden muss.

Zusammengefasst lauten die Anforderungen an die Interaktive 3D-Visualisierung:

- Variabler Animationsablauf, der durch Benutzereingaben bestimmt wird
- Lauffähigkeit auf einem Büro-PC
- Polygonarme Darstellung
- Einfache Benutzeroberfläche, ähnlich mit einigen Bedienelementen der SVP
- Kostengünstige Produktion

Aus den oben genannten Rahmenbedingungen heraus wurde diese Benutzeroberfläche entwickelt:

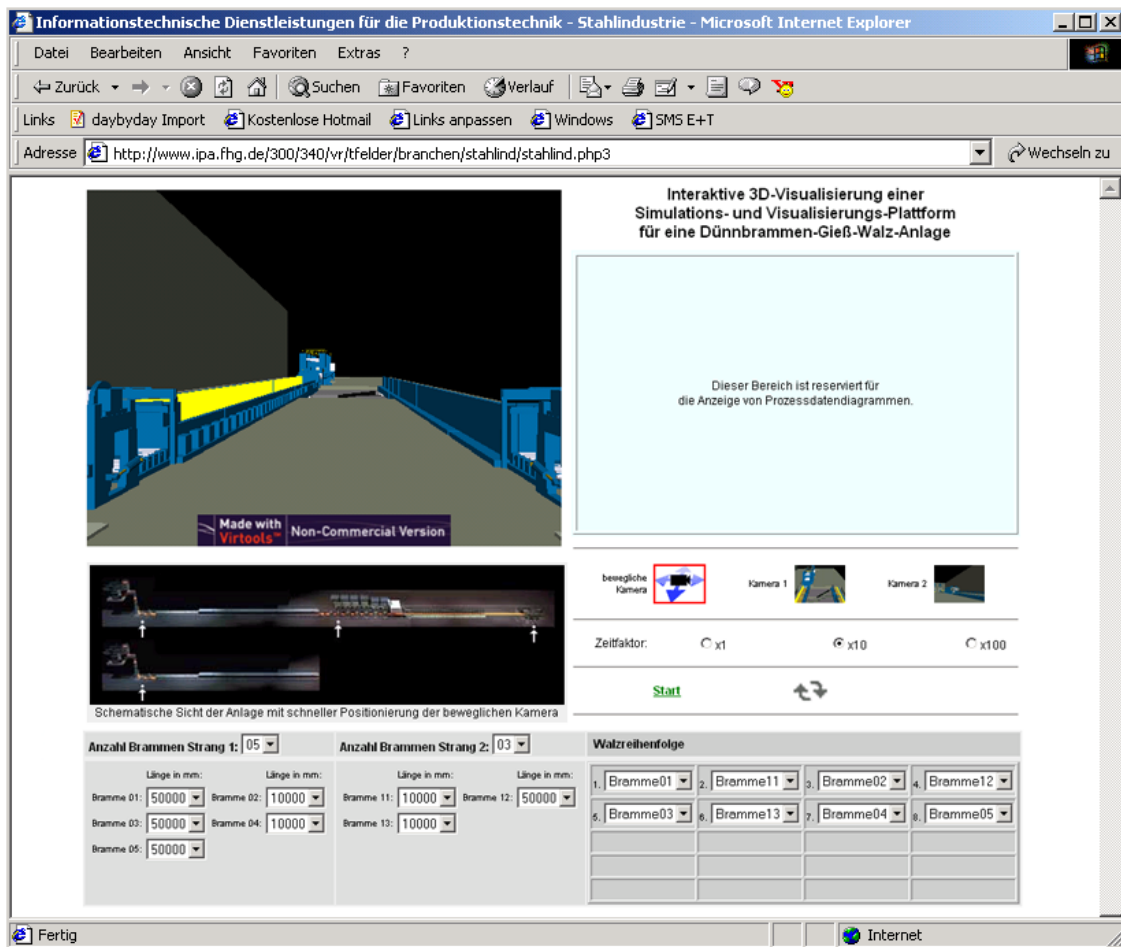


Bild 5: Benutzeroberfläche der interaktiven 3D-Visualisierung

Die Benutzeroberfläche enthält die folgenden Bedienelemente:

- Kamera-Modus
Hier kann der Benutzer wählen, ob er die Kamera frei umher bewegen möchte oder durch schwenkbare, an interessanten Punkten „festinstallierte“ Kameras blicken möchte.
- Start
Startet die Interaktive 3D-Visualisierung
- Reset
Setzt die Interaktive 3D-Visualisierung auf den Anfangszustand zurück
- Zeit-Faktor
Mit der Auswahl eines Zeitfaktors kann die Ablaufgeschwindigkeit eingestellt werden. Der Wert 1 entspricht der Realzeit, bei Auswahl des Faktors 10 oder 100 läuft der Gieß-Walz-Prozess um den entsprechenden Faktor schneller ab.
- 2D-Navigator (Schematische Sicht der Anlage mit schneller Positionierung der beweglichen Kamera)

Hier kann der Benutzer die bewegliche Kamera schnell an interessanten Punkten in der Anlage positionieren.

- 3D-Fenster
Der Benutzer kann die Kamera mit der Maus schwenken (bei gedrückter linker Maustaste). Im „Bewegliche Kamera“-Modus lässt sich die Kamera mittels der Cursor-Tasten umher bewegen. Sie befindet sich dabei ca. 2 m über dem Boden.
Mittels Klick auf Objekte werden Prozessdaten angezeigt.
- Anzahl für Brammen und Walzreihenfolge
Hier kann der Benutzer Anzahl und Längen der Brammen sowie ihre Auftragsreihenfolge bestimmen.

2 Die Produktion

2.1 Auswahl und Bewertung von 3D-Autorenwerkzeugen

2.1.1 Kriterien für die Auswahl

Seitens des „Auftraggebers“ für die interaktive 3D-Visualisierung, des IPA, gab es im wesentlichen zwei Vorgaben:

- Die 3D-Modelle der Anlagenteile sind vertraulich. Es darf also keine Möglichkeit für den Besucher der Webseite geben, die 3D-Modelle herunter laden und in irgendeiner Form bearbeiten oder weiter verwenden zu können.
- Der Preis für die Anschaffung des 3D-Autoren-Werkzeugs und die laufenden Kosten für den Betrieb der interaktiven 3D-Visualisierung müssen minimiert werden.

Die Interaktive 3D-Visualisierung soll für das IPA nicht unmittelbar – also als Produkt – Umsätze generieren. Anpassungen und/oder Weiterentwicklungen, aber auch die Produktion selbst, müssen daher in einem wirtschaftlich vertretbaren Rahmen liegen. Daher war anzustreben, ein 3D-Autorenwerkzeug zu finden, das relativ leicht erlernbar ist. Generell muss dabei in Betracht gezogen werden, dass ein Verzicht auf eine Programmiersprache (und somit ein geringerer Lernaufwand) auch die Flexibilität bei der Produktion beschneidet.

3D-Autorenwerkzeuge können – zunächst ohne Bezug zu einem konkreten Projekt – nach verschiedenen Kriterien eingeordnet werden. Eine erste Unterscheidung nach Autor und Rezipient/Benutzer ist hilfreich.

Aus Sicht des Besuchers von Web-Seiten ist die technische Art und Weise, wie die Szene „erfahrbar“ gemacht wird, von Bedeutung. Für interaktive 3D-Inhalte kommen grundsätzlich zwei Möglichkeiten in Frage: Darstellung der 3D-Szene und zusätzliche Interaktionsmöglichkeiten basierend

Die Produktion

Auswahl und Bewertung von 3D-Autorenwerkzeugen

- auf dem Standard VRML97 bzw. künftig X3D in Verbindung mit Java oder
- auf der Verwendung einer proprietären Technologie mittels eines Browser-PlugIns.

Generell ist jedes zusätzliche PlugIn, das nötig ist, um spezielle Medienformate auf einer WWW-Seite anzuzeigen, dem Besucher ein Dorn im Auge.⁹ Das PlugIn muss installiert und gelegentlich aktualisiert werden und kann eine potenzielle Gefährdung der Datensicherheit darstellen. Hinzu kommen eventuelle Unverträglichkeiten mit anderen Browser- und Systemkomponenten. Die Systemstabilität kann also unter Umständen leiden.

Der technisch interessierte und anspruchsvolle Rezipient kann daran interessiert sein, ob der Renderer auf das plattformübergreifende OpenGL, auf Microsofts *DirectX* bzw. *Direct3D* zurückgreift oder eigene Methoden für eine optimale Berechnungsgeschwindigkeit und somit flüssige Darstellung verwendet. Der Autor muss Rücksicht darauf nehmen, welche technischen Gegebenheiten bei seiner Zielgruppe vorherrschen.

Ein für den 3D-Autor wichtiges Kriterium ist das Maß der technologischen Interaktionsmöglichkeiten, die die 3D-Szene bieten kann. So ist z. B. von Bedeutung, ob die 3D-Szene mit der Webseite, in die sie eingebettet ist, oder mit anderen beteiligten Systemen, etwa einer Datenbank, Informationen austauschen kann. Es stellt sich also die Frage nach Datenschnittstellen.

Desweiteren sind Benutzerschnittstellen von Interesse: Die Verarbeitung von Maus-Signalen ist bei allen untersuchten 3D-Autorenwerkzeugen möglich. Aber können auch Tastatureingaben oder Joystick-Inputs verarbeitet werden? Für Charaktere ist von Bedeutung, ob Sprachausgabe unterstützt wird, ob also eine Lippensynchronizität erreichbar ist.

Weiterhin kann es wünschenswert sein, andere Medientypen wie Video und Audio in die 3D-Szene einzubinden, wie es auch bei allen gängigen professionellen 3D-Programmen möglich ist.

Für wissenschaftliche Visualisierungen, bei denen es auf Präzision ankommt, ist das Echtzeitverhalten ein wichtiges Kriterium: Können eingehende Daten sofort

⁹ Die Anzeige von VRML-Szenen erfolgt zwar auch mittels eines Browser-PlugIns. Dieses wird aber i. d. R., zumindest bei den zwei führenden Browsern Microsoft Explorer und Netscape Navigator, bereits mitgeliefert.

Die Produktion

Auswahl und Bewertung von 3D-Autorenwerkzeugen

verarbeitet werden oder gerät das System ins Stocken oder „überspringt“ Datenwerte bei der Verarbeitung?

Echtzeitfähigkeit oder zumindest Beinahe-Echtzeitfähigkeit ist auch für Online-Spiele von Bedeutung, insbesondere bei Mehrbenutzerumgebungen, bei denen von einander entfernte Spieler gegeneinander antreten können.

Ein bereits im Abschnitt 1.5.2 besprochenes Kriterium ist die Frage nach der Flexibilität bei der Produktion, die durch eine Programmiersprache ermöglicht wird. Wenn eine interaktive 3D-Szene mehr Funktionalität bereit stellen soll, als z. B. ein Verkaufsprodukt am Bildschirm von allen Seiten betrachten zu können, dann kann sie leicht so komplex werden, dass dieser Anforderung nur mit einer Programmiersprache beizukommen ist.

Schließlich kann es unerwünscht sein, dass der Besucher einer Webseite die 3D-Modelle herunter lädt und für seine Zwecke verwendet. Ein angenehmer Nebeneffekt eines proprietären 3D-Formats ist, dass die 3D-Modelle oft in einem nicht verwertbaren Format vorliegen. Sie sind dann nicht herunter zu laden, weil sie nicht im Zugriffsbereich eines Browsers liegen oder zusammen mit anderen Daten wie Programmcode in einer solitären Datei enthalten sind.¹⁰

Zusammenfassend lauten die Kriterien:

- VRML oder proprietäres 3D-Format bzw. Standard oder nicht Standard
- Datenschnittstellen
- Benutzerschnittstellen
- Unterstützung von OpenGL und/oder DirectX
- Medienintegration
- Echtzeitfähigkeit
- Mehrbenutzerfähigkeit
- Programmiersprache
- Schutz der 3D-Modelle

¹⁰ Es ist vielleicht für einen Programmierer des Unternehmens, das das 3D-Autorenwerkzeug herstellt, möglich, die 3D-Objekte aus der kompilierten Szene zu isolieren. Das würde aber sehr unredliche Absichten voraussetzen.

Die Produktion

Auswahl und Bewertung von 3D-Autorenwerkzeugen

Für die interaktive 3D-Visualisierung sind folgende Kriterien von Bedeutung:

- VRML oder proprietäres 3D-Format bzw. Standard oder non-Standard
Vorgabe: nicht VRML
- Datenschnittstellen:
Möglichst JavaScript zum Datenaustausch zwischen WWW-Seite und 3D-Szene, da JavaScript ein vom W3C verabschiedeter Standard ist. Ideal für künftige Anwendungen ist eine Datenbank-Schnittstelle.
- Benutzerschnittstellen:
Maus und Tastatur, wie in der SVP
- Echtzeitfähigkeit
Animationen müssen ähnlich wie in der SVP zeit- und ereignisgesteuert werden können.
- Programmiersprache:
Im Abschnitt 1.5.2 wurde die Anforderung „Variabler Animationsablauf“ besprochen. Um diese Anforderung erfüllen zu können, ist eine Programmiersprache notwendig, da nur mit ihr Abhängigkeiten im Animationsablauf abgebildet und verarbeitet werden können.
- Schutz der 3D-Modelle
Die Vorgabe „nicht VRML“ impliziert den Schutz der 3D-Modelle, solange das eingesetzte 3D-Autorenwerkzeug ein proprietäres Format verwendet, das von anderen Programmen, die insbesondere die Modellierung und Formatumwandlung in gängige Formate unterstützen, nicht gelesen werden kann.

Die Kriterien „Standard oder nicht“ und Daten- und Benutzerschnittstellen sind relativ leicht zu prüfen. Die meisten Hersteller von 3D-Autorenwerkzeugen bieten eine zeitlich beschränkte, voll funktionsfähige Demo-Version ihres Produkts zum Download über das WWW an. Oft ist eine vollständige oder wenigstens teilweise Dokumentation dabei, so dass sich zumindest die technischen Fähigkeiten des Produkts ermitteln lassen. Desweiteren finden sich auf der Homepage des Herstellers fast immer Links zu Webseiten, die Ergebnisse seines Produkts vorstellen.

Etwas schwieriger stellt sich hingegen die Überprüfung der Echtzeitfähigkeit und der Programmiersprache dar. Der Leistungsumfang der Programmiersprache kann noch in etwa an der Dokumentation gemessen werden. Bei der Recherche fanden sich jedoch nur wenige, dazu kleine (im Sinne von Polygonzahl und Anzeigegröße) Beispiele oder Projekte, bei denen es auf die Echtzeitfähigkeit ankam.

In den folgenden Abschnitten werden die untersuchten 3D-Autorenwerkzeuge kurz vorgestellt und hinsichtlich der oben entwickelten und für das Projekt relevanten Kriterien bewertet.

Leider war eine der zentralen Anforderungen an die interaktive 3D-Visualisierung die nach dem Schutz der 3D-Modelle, so dass viele 3D-Autorenwerkzeuge von vorneherein nicht in Frage kamen, weil sie auf VRML-Modellen basieren, und daher nicht eingehender untersucht wurden. VRML-Dateien können zwar komprimiert auf dem WWW-Server abgelegt werden. Da sie aber zip-komprimiert sind, ist es mit geringem Aufwand möglich, die Modelle „von Hand“ herunter zu laden, zu dekomprimieren und damit zu arbeiten.

Einige VRML-basierte 3D-Autorenwerkzeuge sollen dennoch kurz vorgestellt werden, um dem interessierten Leser einen groben Überblick über Produkte und Leistungsumfang zu verschaffen. Es wird kein Anspruch auf Vollständigkeit erhoben. Die Beschreibungen basieren auf dem Stand der Untersuchungen im Winter 2000/2001.

2.1.2 Eyematic Shout3D 2 Beta 4 und Virtock Spazz3D Version 2.4 c

Das Produkt *Shout3D* des US-amerikanischen Unternehmens Eyematic basiert auf VRML und besteht aus dem *Shout3DWizard* und 144 Java-Klassen. *Shout3D* ist für Windows und Mac erhältlich. Die Komponente zum Anzeigen der 3D-Szene ist in Java realisiert und wird beim Besuch der die 3D-Szene enthaltenden Seite zusammen mit weiteren benötigten Java-Komponenten vom WWW-Server herunter geladen.

Der *Shout3DWizard* dient im wesentlichen zur Erstellung einer HTML-Rahmenseite, in deren Code die 3D-Szene als Java-Objekt eingebettet ist, sowie für Vorschauzwecke.

Die Produktion

Auswahl und Bewertung von 3D-Autorenwerkzeugen

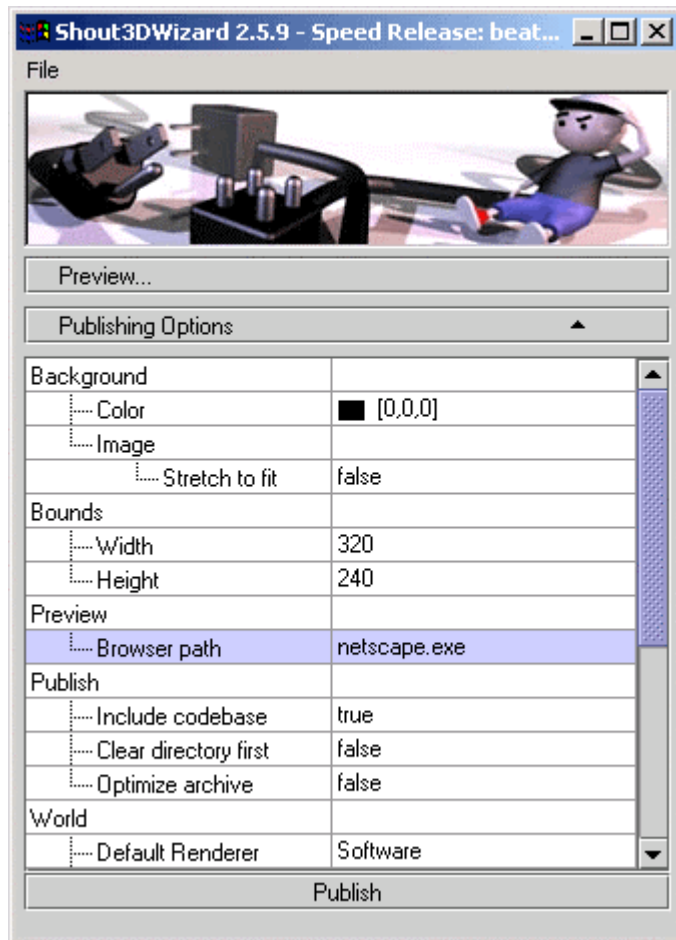


Bild 6: Benutzeroberfläche des Shout3DWizard

Shout3D bietet eine JavaScript-Schnittstelle, die für einfache Objektpräsentationen ausreicht, so dass Interaktionen innerhalb des HTML-Codes realisierbar sind. Weitergehende Funktionen müssen mit Java realisiert werden.

Shout3D beherrscht *Threaded Loading*, d. h. der Benutzer der 3D-Szene muss nicht tatenlos warten, bis alle Daten herunter geladen wurden. Die Szene kann starten, sobald alle 3D-Objekte verfügbar sind. Im Hintergrund werden fehlende Elemente wie Texturen nach und nach herunter geladen und so wird die 3D-Szene vervollständigt.

Autorenfunktionen im eigentlichen Sinne bietet *Shout3D* nicht. Solche Funktionen in Java oder VRML „von Hand“ zu realisieren, wäre sehr mühsam. Für diese Zwecke bietet Virtock Technologies Inc. das Produkt *Spazz3D* an. Hiermit können VRML-Szenen importiert oder weitere Primitive erzeugt und bearbeitet werden. Umfangreiche sogenannte *Wizards* helfen bei der Erstellung von 3D-Text, Animationen, animierten Kameras und Head Up Displays.

Die Produktion

Auswahl und Bewertung von 3D-Autorenwerkzeugen

Spazz3D besitzt zwei spezielle Komponenten. Die erste Komponente ist eine Schnittstelle zu *Shout3D*. Sie ermöglicht die Übergabe der mit *Spazz3D* erstellten 3D-Szene an *Shout3D*. Somit wird die 3D-Szene bequem in eine HTML-Umgebung eingebettet.

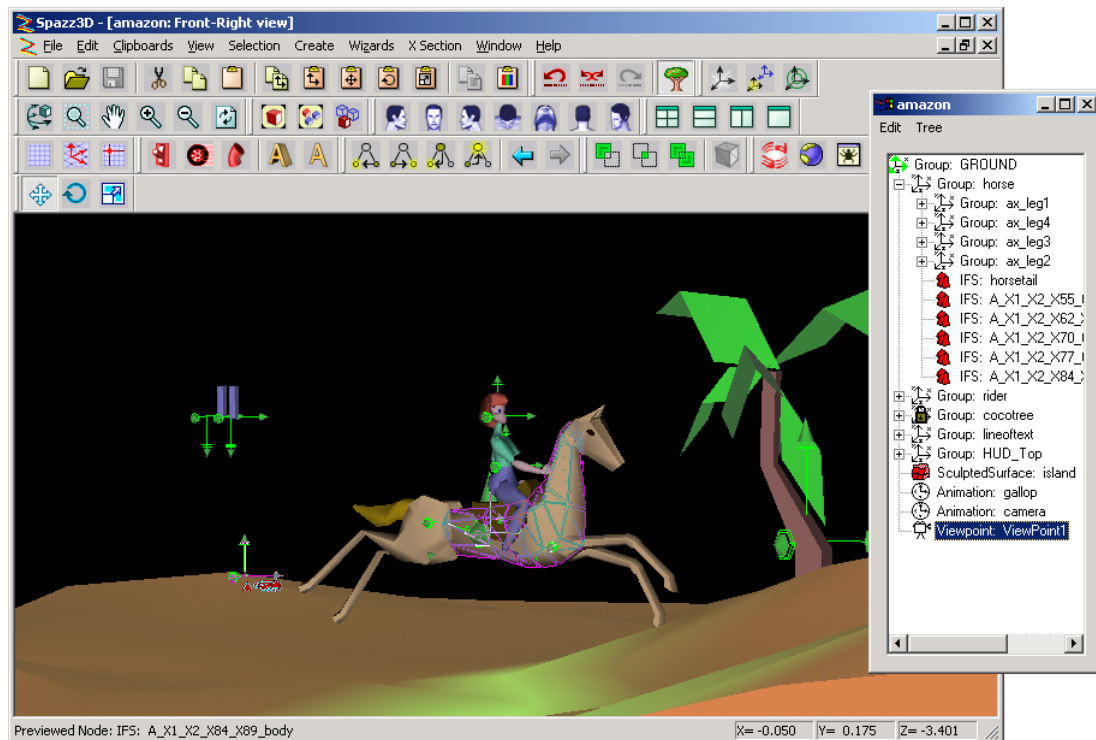


Bild 7: Benutzeroberfläche von *Spazz3D*

Die zweite Komponente ist ein Assistent für Manipulationen von Avataren¹¹, der einen von dem Unternehmen blaxxun empfohlenen Standard für Avatar-Gesten unterstützt. blaxxun ist ein engagiertes Mitglied des Web3D Consortiums.

Spazz3D beherrscht lediglich den Import von VRML-Modellen. In der vorliegenden Version hatte *Spazz3D* Schwierigkeiten beim Import größerer, d. h. polygonreicher Modelle. Die Anwendung reagierte dann nicht mehr.

¹¹ Eine Avatar ist die virtuelle Darstellung des Benutzers. Er befindet sich am Beobachtungspunkt, von dem aus der Benutzer die Szene sieht. Bewegt sich der Benutzer allein durch die Szene, dann dient der Avatar nur dazu, Kollisionen des Benutzers mit Objekten der Welt festzustellen. In einer Mehrbenutzerwelt jedoch legt der Avatar auch fest, wie ein Benutzer von anderen Benutzern gesehen wird. Standards für diese und ähnliche Probleme werden in Arbeitsgruppen des Ende 1996 gegründeten VRML-Konsortiums ausgearbeitet. (Quelle: Informatik-Lexikon der Gesellschaft für Informatik e. V., www.gi-ev.de/informatik/lexikon/inf-lex-vrml.shtml)

Ursprünglich bezeichnet das Wort Avatar eine irdische Inkarnation einer hinduistischen Gottheit, Quelle: Online Lexikon der Tageszeitung „Die Welt“)

2.1.3 Parallel Graphics: Internet Scene Assembler (ISA) v1.0

Das Unternehmen Parallel Graphics mit Hauptsitz in Irland bietet den VRML-basierten *Internet Scene Assembler* zur Produktion von interaktiven 3D-Szenen an.

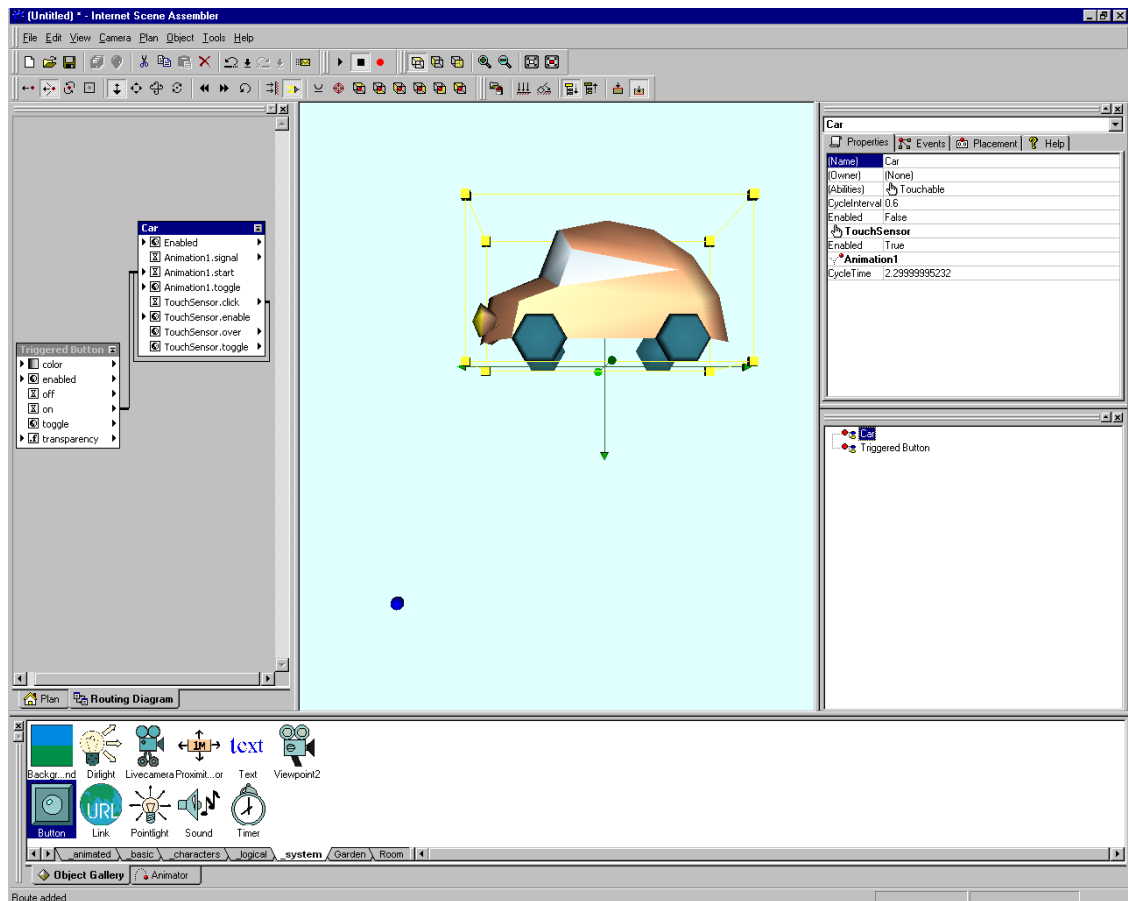


Bild 8: Benutzeroberfläche des *Internet Scene Assembler*

Der ISA bietet eine für Autorenwerkzeuge typische Diagramm-Darstellung von Abhängigkeiten zwischen Objekten, ihren Aktionen und Ereignissen, die Aktionen auslösen können. Beim ISA heißen diese Diagramme *Routing Diagrams*. Solche Diagramme haben einen oder mehrere Eingänge und Ausgänge. Die grafische Verknüpfung eines Eingangs eines Objekts A mit einem Ausgang eines anderen Objekts B oder desselben Objekts A bedeutet i. d. R. die Aktivierung einer Animation des Objekts A. Auf diese Art können komplexe Animationen und Interaktionen hergestellt werden.¹²

¹² Diese Art der visuellen Programmierung wird in Abschnitt 2.2.3 anhand projektbezogener Beispiele näher erläutert.

Die Produktion

Auswahl und Bewertung von 3D-Autorenwerkzeugen

Die Benutzeroberfläche ist aufgeteilt in verschiedene Fensteransichten:

- Plan:
Eine zweidimensionale schematische Draufsicht auf die Szene.
- Routing Diagram:
Das Fenster zur Programmierung der Animationen und Interaktionen
- Properties:
Eigenschaften von Objekten
- Events:
Listendarstellung von Ereignissen und ihren Verknüpfungen mit Aktionen
- Placement:
Eingabefelder zur numerischen Manipulation von Objektpositionen und -ausrichtungen
- Help
Hilfetexte mit Erläuterungen zu vordefinierten Objekten und ihren Eigenschaften und auslösbaren Ereignissen
- Ein unbenanntes Fenster mit einer hierarchischen Objektliste
- Object Gallery
Eine Auswahl von Objekten, einigen logischen Operatoren und speziellen Objekten wie URLs, Knöpfen oder Timern
- Animator
Ein Fenster mit einer Zeitschiene zur Erstellung von Objektanimationen

ISA kann VRML-Dateien und *ISB object files* importieren. *ISB object files* werden mit dem *Internet Space Builder (ISB)* erstellt. Der *ISB* ist ebenfalls ein Produkt von Parallel Graphics und dient zum Modellieren und Texturieren von 3D-Szenen.

Für Demonstrationszwecke kann die Videoaufzeichnungsfunktion von Interesse sein, die eine Video-Datei im AVI-Format produziert.

In der vorliegenden Version war *ISA* ziemlich instabil. Das Programm ist häufig unvermittelt abgestürzt.

2.1.4 Cryonetworks: Site Construction Set (SCS)

Die in Abschnitt 2.1.1 aufgeführten Kriterien hätte das Produkt *Site Construction Set* des französischen Herstellers Cryonetworks wahrscheinlich erfüllt. Zwei Gründe führten leider dazu, dass SCS nicht in die engere Auswahl kam:

- Trotz aller Bemühungen des Autors und des technischen Supports von Cryonetworks konnte die angeforderte Demo-Version nicht installiert werden. Versucht wurde dies auf verschiedenen Rechnern unter Windows NT 4 und Windows 2000.
- Zum Vorhalten von SCS-Szenen auf dem WWW-Server wird eine Server-Software benötigt. Die für die Server-Software anfallenden Lizenzgebühren betragen mehrere hundert DM pro Jahr. Das wäre zu teuer.

SCS kommt laut Herstellerangaben aufgrund seiner Mehrbenutzerfähigkeit vorwiegend zur Produktion von E-Shops, E-Learning-Umgebungen, Online-Spielen und –Communities zum Einsatz.

Für einfache Projekte bietet SCS eine grafische Autorenumgebung, in der sich per „Drag & Drop“ 3D-Objekte und Funktionen platzieren und miteinander verknüpfen lassen.

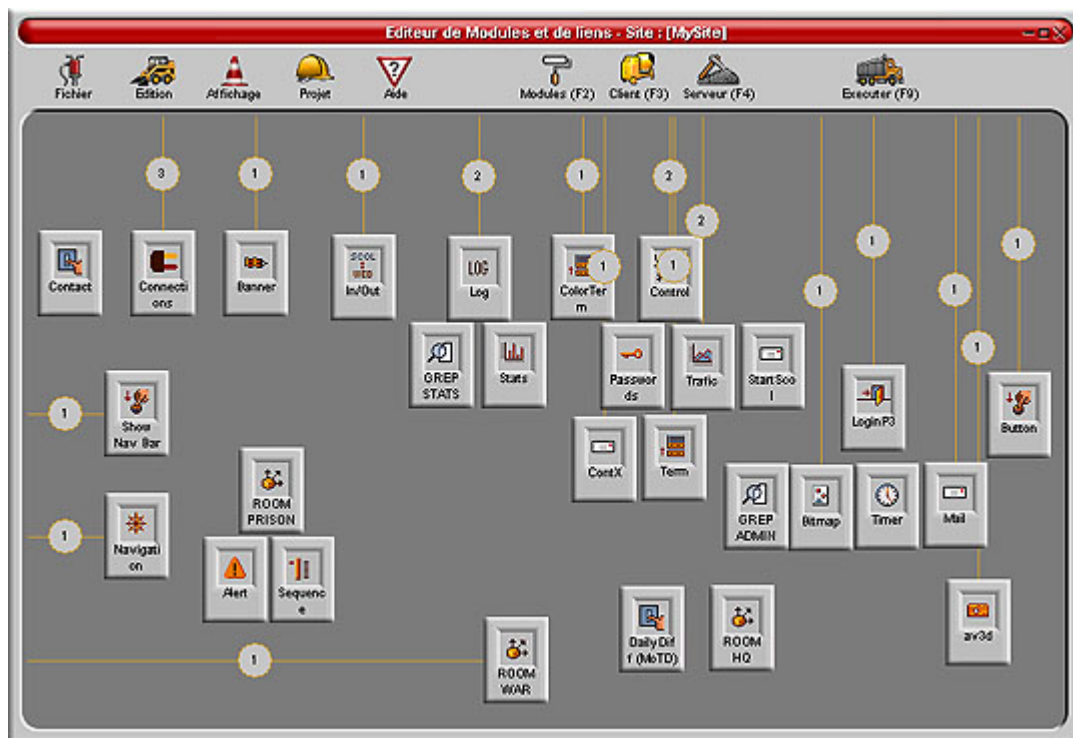


Bild 9: Benutzeroberfläche von Site Construction Set

Die Produktion

Auswahl und Bewertung von 3D-Autorenwerkzeugen

Für anspruchsvollere Projekte steht die C-ähnliche Programmiersprache SCOL (Standard Cryo Online Language) zur Verfügung. Mit ihr lassen sich individuelle Funktionalitäten realisieren. Besonders interessant ist dabei die SQL-Schnittstelle zu relationalen Datenbanken.

Desweiteren soll SCS laut Herstellerangaben aufgrund seiner offenen Systemarchitektur kompatibel sein zu den Betriebssystemen Microsoft Windows, Unix, Linux und Mac OS. Außerdem unterstütze es die Integration verschiedener Medien wie WebCam, RealVideo von Real und QuickTime von Apple.

Das Abspielen der 3D-Szenen erfolgt durch ein Browser-PlugIn, das der Besucher der Webseite zuvor auf seinem Rechner installieren muss.

2.1.5 Nemo: NeMo Creation 1.0.1

Vom französischen Hersteller Virtools S. A. (ehemals Nemo) stammt das Produkt *NeMo*, das es zum Zeitpunkt der Untersuchung in zwei Versionen gab: *Creation* und *Dev*. Die primäre Zielgruppe von *NeMo* sind Spieleentwickler.

Dev unterscheidet sich von *Creation* durch eine zusätzliche Entwicklungsumgebung und der Funktion zur Generierung von Szenen als ausführbare Programme, für die kein Browser-PlugIn benötigt wird.

Mit *Dev* können unter Verwendung einer Programmiersprache wie C++ individuelle Funktionen in Form von DLLs entwickelt werden, die sich als sog. *Building Blocks* in die Autorenumgebung von *NeMo* einbinden lassen.

Zur Untersuchung stand eine 90-Tage-Demo-Version¹³ von *NeMo Creation* zur Verfügung.

¹³ Eine Demo-Version, die 90 Tage lang uneingeschränkt funktioniert, ist recht großzügig. Interessanterweise bietet Virtools seit der Einführung von *Virtools Dev 2.0* über seine Webseite überhaupt keine Demo-Version mehr an.

Die Produktion

Auswahl und Bewertung von 3D-Autorenwerkzeugen

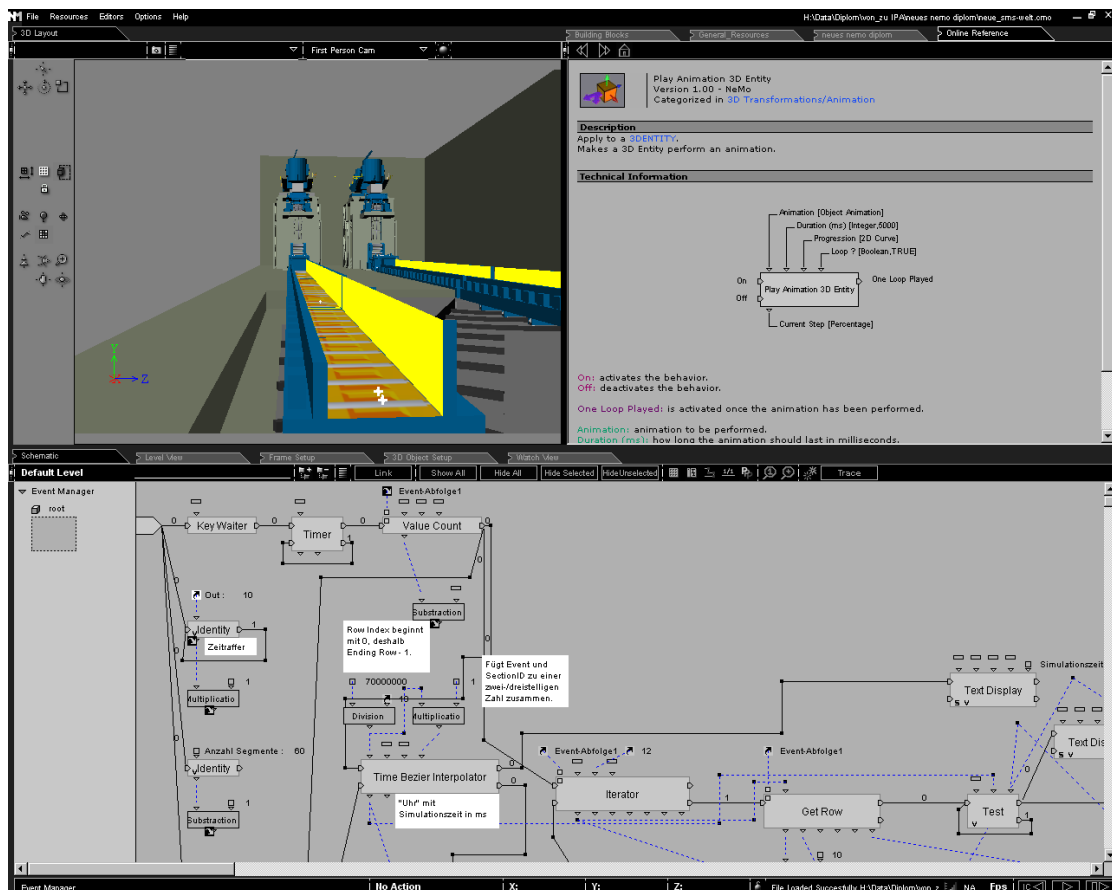


Bild 10: Benutzeroberfläche von NeMo Creation 1.0.1

Die Benutzeroberfläche von NeMo stellt sich dreigeteilt dar:

- Links oben befindet sich die 3D-Ansicht, in der sich neben einer perspektivischen Ansicht die für 3D-Programme gängigen Drauf- und Seitenansichten sowie alle in der Szene befindlichen Kameras einstellen lassen.
- Rechts oben ist ein Bereich für Ressourcen. Dazu gehören die sog. Building Blocks, die unten kurz und im Abschnitt 2.2.3 ausführlicher erläutert werden. Desweiteren gehören zu den Ressourcen eine Liste aller in der Szene enthaltenen 3D-Modelle und die Funktionsbeschreibungen der Building Blocks.
- Die untere Bildhälfte zeigt wahlweise
 - Skripte (in Bild 10 ist der Ausschnitt eines Skripts zu sehen),
 - eine hierarchische Listenansicht aller in der Szene verwendeten Objekte und Skripte,
 - die sog. Watch View zur Verfolgung von Werten während der Vorschau einer Szene oder
 - Eigenschaften eines ausgewählten Objekts.

Die Produktion

Auswahl und Bewertung von 3D-Autorenwerkzeugen

Erstellte Szenen können im Browser mittels eines zu installierenden PlugIns oder in einem eigenständigen Programm abgespielt werden. Beide Darstellungsarten unterstützen OpenGL.

Für den Datenaustausch zwischen 3D-Szene und Browser steht dem 3D-Autor eine JavaScript-Schnittstelle zur Verfügung. Mit dieser Schnittstelle lassen sich allerdings nur in sehr begrenztem Umfang Daten austauschen. Näheres dazu im Abschnitt 2.2.4.

NeMo importiert das 3D-Format *3ds* für Modelle von *3D Studio Max*, das hausinterne *nmo*-Format und das *x*-Format von Microsoft *Direct3D*. Für andere 3D-Programme wie *Maya* und *SoftImage* gibt es entsprechende Export-PlugIns, die Modelle im *nmo*-Format ausgeben. Die erstellten Szenen in Form einer *cmo*-Datei sind komprimiert und enthalten die 3D-Modelle sowie ihre Animationen, Texturen, Materialien und Skripte zusammen. Sie lassen sich mit *NeMo* nicht mehr laden.

Es steht eine Programmiersprache in Form grafischer Skripte zur Verfügung. Die von Virtools vorgefertigten 307 Building Blocks und ihre Parameter sind die kleinsten Elemente dieser Skripte. Für die interaktive 3D-Visualisierung war abzu-sehen, dass das ausreichend sein würde, denn die Building Blocks umfassen einfache Rechenoperationen ebenso wie komplexe Funktionen, z. B. für LOD und Funktionen zur Tabellenverwaltung, zur Transformation von 3D-Objekten oder zur Steuerung von Charakteren.

Einen bedeutenden Schwachpunkt stellt die mitgelieferte Dokumentation dar. Das 125-seitige Handbuch beschreibt nur die wichtigsten Konzepte und führt mit einem konkreten Beispiel in einige einfache Animations- und Programmierungstechniken ein. Die etwas ausführlichere Online-Dokumentation, auf die innerhalb von *NeMo* zugegriffen werden kann, bietet keine vollständige Referenz, wie sie z. B. bei den Dokumentationen von *3D Studio Max* oder *Macromedia Director* zu finden sind. Aus diesem Grund enthält Anlage A eine alphabetisch sortierte Liste der *Building Blocks* und ihrer Kategorien, die das Auffinden von *Building Blocks* in *NeMo* erleichtert.

Somit war *NeMo* jedoch - neben SCS, das nicht praktisch untersucht werden konnte - das einzige 3D-Autorenwerkzeug, das alle im Abschnitt 2.1.1 genannten Kriterien erfüllt. Im folgenden Kapitel wird gezeigt, welche dieser Kriterien *NeMo Creation* im praktischen Einsatz besser oder schlechter erfüllt hat.

Die Produktion

Auswahl und Bewertung von 3D-Autorenwerkzeugen

Während der Entstehung dieser Arbeit änderte das Unternehmen Nemo nicht nur seinen Namen in Virtools, sondern auch seine Produktpolitik und –palette. Heute gibt es das 3D-Autorenwerkzeug mit dem neuen Namen *Virtools* nur noch in der Version *Dev 2.0* zu einem wesentlich höheren Preis, der den Ausschlag dazu gab, für die Erstellung dieser Arbeit noch *NeMo Creation 1.0.1* anzuschaffen.¹⁴ Die Beschaffung war mit einer erheblichen Verzögerung von drei Monaten verbunden, weil die Genehmigung für die Anschaffung durch das Institut erst erfolgte, als Virtools bereits den Vertrieb von NeMo eingestellt hatte. Nach Gesprächen mit dem Vertriebsbeauftragten von Virtools erklärte dieser sich aber bereit, eine Ausnahme zu machen.

Angeschafft wurde die „Non-Commercial-“ bzw. „Educational Version“, die sich von der regulären Version in zwei Punkten unterscheidet:

- Der Verkaufspreis ist halb so hoch wie für die reguläre Version.
- Die erstellten Szenen tragen beim Abspielen im Browser am unteren Rand des Szenenbilds folgendes „Branding“, das die Erstellung der Szene durch die „Educational Version“ kennzeichnet:

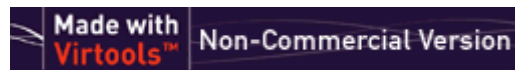


Bild 11: „Branding“ der „Educational Version“ von NeMo Creation

2.1.6 Übersicht der untersuchten Produkte

Untersucht wurden 3D-Autorenwerkzeuge, die zur Produktion interaktiver 3D-Szenen eingesetzt werden können. Die nachfolgende Tabelle zeigt, welche der Produkte die Anforderungen für die Interaktive 3D-Visualisierung - soweit das mit den Demo-Versionen festzustellen war -, erfüllen. Die Zusammenstellung der untersuchten Produkte erfolgte anhand von Recherchen im WWW.

¹⁴ Die Non-Commercial-Lizenz von Dev 1.0.1 wurde für 3.890,- DM angeboten, die von Creation für 950 DM. Heute kostet Dev 2.0 ca. 4.000 €.

Die Produktion

Eine Schritt-für-Schritt-Anleitung

	3D-Format		Daten-schnittstellen	Benutzer-schnittstellen	Echtzeitfähig	Programmiersprache	Schutz der 3D-Modelle
	VRML	Proprietär					
Eyematic Shout3D 2 Beta 4 und Virtock Spazz3D Version 2.4 c	x	-	JavaScript	Maus, Tastatur	nicht bekannt	nein	Nein
Parallel Graphics Internet Scene Assembler	x	-	Keine	Maus, Tastatur	nicht bekannt	nein	nein
v1.0 Cryonetworks Site Construction Set	-	x	SQL	Maus, Tastatur	Ja, laut Herstellerangaben	ja	ja
Nemo NeMo Creation 1.0.1	-	x	JavaScript	Maus, Tastatur	Ja, eingeschränkt (vgl. Abschnitt 2.2.3)	ja (grafische Skripte)	ja

Tabelle 1: Übersicht der untersuchten Produkte hinsichtlich der Anforderungen an die Interaktive 3D-Visualisierung

2.2 Eine Schritt-für-Schritt-Anleitung

2.2.1 Die 3D-Modelle - von *Medit* zu *NeMo*

In diesem Abschnitt wird beschrieben, welcher Konvertierungspfad eingeschlagen und welche Möglichkeiten ausgeschöpft wurden, um die endgültige interaktive 3D-Szene so klein wie möglich werden zu lassen. Von Bedeutung ist hierbei die Anzahl der Modelle, Polygone und Materialien in der Szene.

Nach der Polygon- und Materialienreduzierung wurden die einzelnen Modelle sukzessive in *3D Studio Max* in ihre endgültige Position gebracht.

Das „Rohmaterial“ für die Interaktive 3D-Visualisierung - die 3D-Modelle sämtlicher Anlagenteile - lag bereits voll ausmodelliert, teilweise texturiert und größtenteils in mehreren LOD im *medit*-Format vor.

Für den Import in *NeMo* wurde sein eigenes *nmo*-Format gewählt, um sicher zu stellen, dass alle Modelle gleichermaßen den *NeMo*-Erfordernissen entsprechen. Zur Erzeugung des *nmo*-Formats liegt *NeMo* ein PlugIn für *3D Studio Max* bei.

Die Produktion

Eine Schritt-für-Schritt-Anleitung

Die *Medit*-Modelle mussten also in *NeMo*-Modelle überführt werden. Das Programm *Medit* kann folgende Formate exportieren:

- DXF
- IRIS Inventor
- Multigen .flt
- VR4
- VRML 1.0
- Wavefront obj

Darunter befinden sich zwei Formate - VRML und obj -, die von den meisten gängigen 3D-Programmen importiert werden können.

Wie eingangs erwähnt, müssen die Polygone der 3D-Modelle stark reduziert werden. *3D Studio Max* verfügt u. a. über eine Funktion zur Polygonreduzierung. Fast genau die gleichen Funktionen, aber eine wesentlich bessere visuelle Rückmeldung über Ergebnisse der Polygonmanipulationen bietet die Software *CosmoWorlds* von Silicon Graphics, die auch zur Verfügung stand. Da *CosmoWorlds* VRML-basiert ist, musste keine zusätzliche Formatkonvertierung vorgenommen werden.

Der Konvertierungspfad sieht also folgendermaßen aus:

- medit-Dateien → *Medit*, Export: VRML-Dateien
- VRML-Dateien → *CosmoWorlds*, Reduktion der Polygone und Export: VRML-Dateien
- VRML-Dateien → *3D Studio Max*, weitere Bearbeitung (s. u.) und Export: Creation-Dateien

Im folgenden werden die einzelnen Arbeitsschritte beschrieben:

Konvertierung der medit-Dateien in VRML-Dateien

Die komplette Anlage ist in 30 medit-Dateien abgelegt. Diese müssen nicht einzeln in *Medit* geladen und im VRML-Format gespeichert werden. Zu *Medit* gehören kommandozeilenorientierte Konvertierungsprogramme. Mittels eines Shell-Skripts lassen sich so bequem alle Modelle in einem Durchlauf konvertieren.

Reduktion der Polygone in *CosmoWorlds*

Diese Aufgabe lässt sich leider nicht automatisieren, sondern benötigt menschliche Entscheidungen. Im „Polygon Reduction Editor“ von *CosmoWorlds* lassen sich mit Schiebereglern vier Faktoren manipulieren:¹⁵

- Oberflächenkrümmung („Delete Points by Curvature“)
Polygonpaare - genauer gesagt: benachbarte Paare von Dreiecksflächen - werden zu einem Polygon vereinfacht, wenn ihr Öffnungs- (auch: V-) Winkel innerhalb eines einstellbaren Grenzwertes zwischen 0 und dem größten Öffnungswinkel, den es im betreffenden Modell bzw. in der Szene gibt, liegt. „Zerklüftete“ Objekte können so „geglättet“ werden.
- Fläche („Discard Triangles by Area“)
Polygone, deren Fläche größer ist als der einstellbare Grenzwert, werden entfernt. Die Werte von 0 bis 1 ermittelt der Editor aus den Flächen der Polygone vom kleinsten bis zum größten Polygon im Modell bzw. in der Szene.
- Kantenlänge („Discard Edges by Length“)
Kanten, die eine bestimmte Länge erreichen, werden entfernt. Die Werte von 0 bis 1 ermittelt der Editor aus den Kantenlängen der Polygone von der kürzesten bis zur längsten Kante im Modell bzw. in der Szene.
- Entfernung zwischen zwei Vertices („Merge Initial coordinates“)
Ist die Entfernung zwischen zwei Vertices kleiner als der eingestellte Grenzwert, werden diese Vertices zu einem Vertex zusammengefasst. Auch hier werden die Werte von 0 bis 1 aus der kürzesten und der längsten Entfernung zweier Vertices im Modell bzw. in der Szene ermittelt.

^v vgl. *CosmoWorlds 1.0.3* Online-Dokumentation

Die Produktion

Eine Schritt-für-Schritt-Anleitung

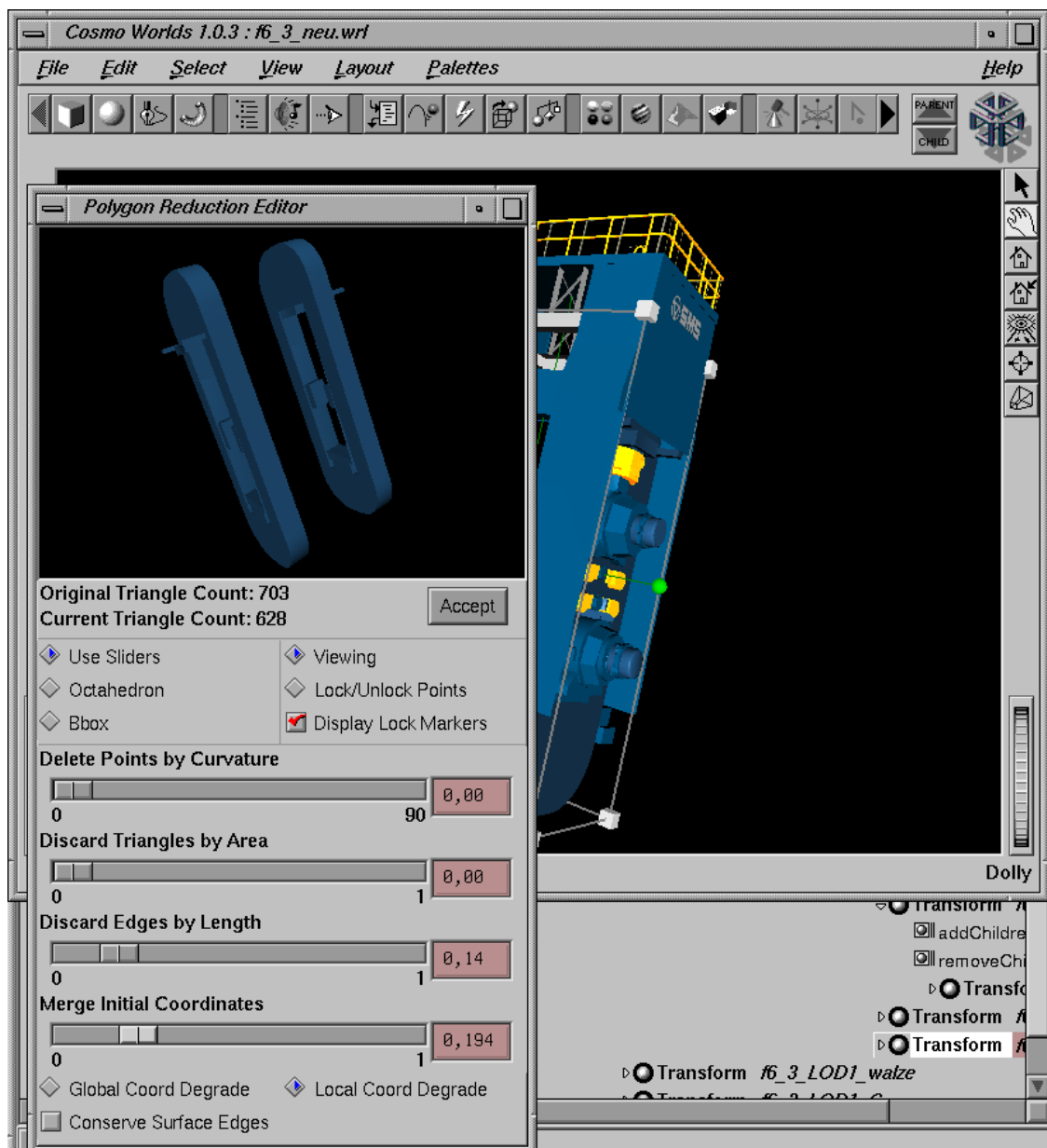


Bild 12: „Polygon Reduction Editor“ von CosmoWorlds

Einzelne Vertices lassen sich mit der Option „Lock/Unlock Points“ für die Reduktion sperren, so dass sie von Reduktionsmanipulationen unbeeinflusst bleiben.

Mit der Auswahl „Global Coord Degrade“ oder „Local Coord Degrade“ wird festgelegt, ob die Ermittlung der Minimal- und Maximalwerte der Schieberegler die Polygone nur des ausgewählten Modells oder der gesamten Szene einbezieht.

Mit der Option „Conserve Surface Edges“ kann erreicht werden, dass auch bei hohen Werten eines Reduktionsfaktors die äußere Form erhalten bleibt.

Die Produktion

Eine Schritt-für-Schritt-Anleitung



Die besten Ergebnisse konnten mit den Faktoren Kantenlänge und Entfernung zwischen zwei Vertices erzielt werden.

Vereinzelte traten Modelle mit doppelten Polygonen auf, d. h. zwei Polygone gleicher Form befanden sich an der gleichen Position und hatten dieselbe Ausrichtung. Während dies in *CosmoWorlds* nicht zu sehen war, verursachten diese doppelten Polygone in der perspektivischen Ansicht und im gerenderten Bild von *3D Studio Max* moirée-ähnliche Muster:

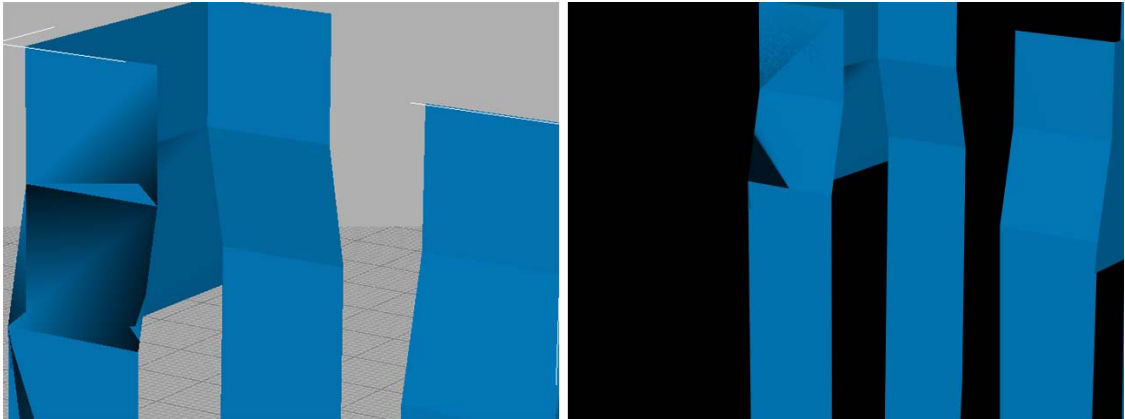


Bild 13: Bildfehler durch doppelte Polygone in der perspektivischen Ansicht (links) und im gerenderten Bild (rechts)

Die Polygonzahl **aller** Modelle wurde so weit wie möglich reduziert, da zu diesem Zeitpunkt noch nicht genau abzusehen war, wie viele Polygone insgesamt wegfallen würden und in welchem Umfang LOD in der Interaktiven 3D-Visualisierung einsetzbar sein würden. Tatsächlich wurden nur die am wenigsten detaillierten LOD verwendet.

Der Zeitaufwand dieser Reduzierungsmaßnahme ist bei der Gesamtzahl von ca. 350.000 Polygonen nicht zu unterschätzen. Im Durchschnitt musste bei diesem Projekt für ein komplettes Anlagenteil mindestens ein Arbeitstag veranschlagt werden.

Auch wenn es im Sinne der Darstellungsqualität gewesen wäre, wurde auf das Neumodellieren einiger einfacherer Modelle aus Zeitgründen und zugunsten des Einsatzes von Texturen verzichtet. So werden z. B. die Geländer der Walztürme und die Rollschienen in den Ofensträngen von Texturen dargestellt.

Die Anlagenteile sind aus mehreren Objekten zusammengesetzt. Die dem VRML-Standard entsprechend hierarchisch aufgebauten Objektbäume enthalten weitere

Die Produktion

Eine Schritt-für-Schritt-Anleitung

sog. Knoten und Gruppenknoten und bilden so Ebenen.¹⁶ Je nach Komplexität sind das 30 oder mehr Ebenen (vgl. Anlage B).

Der Zeitaufwand für die Polygonreduzierung steigt noch, wie bei diesem Projekt, wenn viele einzelne Geometrien in *Medit* keinen Namen erhalten haben. *CosmoWorlds* vergibt dann beim Import automatisch nummerierte Namen, die weder über das Objekt selbst noch über die Position des Objekts innerhalb des Objektbaums etwas aussagen. Es ist dann notwendig, zumindest den Geometrien in *CosmoWorlds* manuell aussagekräftige Namen zu vergeben, da diese später in *3D Studio Max* und in *NeMo* sowie für die JavaScript-Schnittstelle benötigt werden. Für einen besseren Überblick im *Outline Editor* von *CosmoWorlds* empfiehlt es sich, auch einige der Gruppenknoten zu benennen.



Der VRML-Knotenpunkt einer Geometrie heißt *IndexedFaceSet*. Er muss also einen aussagekräftigen Namen erhalten (vgl. Anlage B).



Die verwendete Version von *CosmoWorlds* verwirft Nummern am Ende eines Namens, wenn sie ohne weiteren Text nach einem Unterstrich (_) stehen, z. B. heißt „caster_2_neu_top_LOD1_2“ beim nächsten Öffnen der Datei nur noch „caster_2_neu_top_LOD1“. Daher empfiehlt es sich, beim Nummerieren den Unterstrich zu vermeiden oder einen Text an die Nummer zu hängen, z. B. „..._Teil1“.

Weitere Bearbeitung in *3D Studio Max* und Export nach *NeMo*

Als nächstes wurden die VRML-Dateien in *3D Studio Max* importiert. Hier erfolgte eine weitere Reduktion des Datenumfangs vor allem dadurch, dass überflüssige Objekte entfernt und gleiche Materialien zusammen gefasst wurden (wie im folgenden Abschnitt beschrieben).

Sämtliche VRML-spezifischen Knotenpunkte entfallen, da sie in *3D Studio Max* in *Dummy*-Objekte umgewandelt werden (in *NeMo* würden diese dann in *Frames*¹⁷ umgewandelt) und keinen Zweck mehr erfüllen, dafür aber Speicherplatz belegen. *NeMo* unterstützt keine VRML-Knoten.

¹⁶ Vgl. Hase, Hans-Lothar: Dynamische virtuelle Welten mit VRML 2.0

¹⁷ In *NeMo* ist der Begriff *Frame* verwirrenderweise doppelt belegt. Er bezeichnet sowohl sog. *3D-Entities*, also 3D-Objekte ohne Geometrien, die lediglich eine Position im Koordinatensystem und eine Ausrichtung besitzen, als auch eine Renderperiode (vgl. Abschnitt 2.2.3).

Die Produktion

Eine Schritt-für-Schritt-Anleitung



Die *Dummy*-Objekte können bequem entfernt werden, indem die betreffenden Objektbäume mit dem Selektionswerkzeug ausgewählt, die Option „Helpers“ deaktiviert und anschließend die Selektion umgekehrt wird. Die verbleibende Auswahl enthält alle zu löschenden *Dummy*-Objekte. Zurück bleiben die benötigten 3D-Modelle.

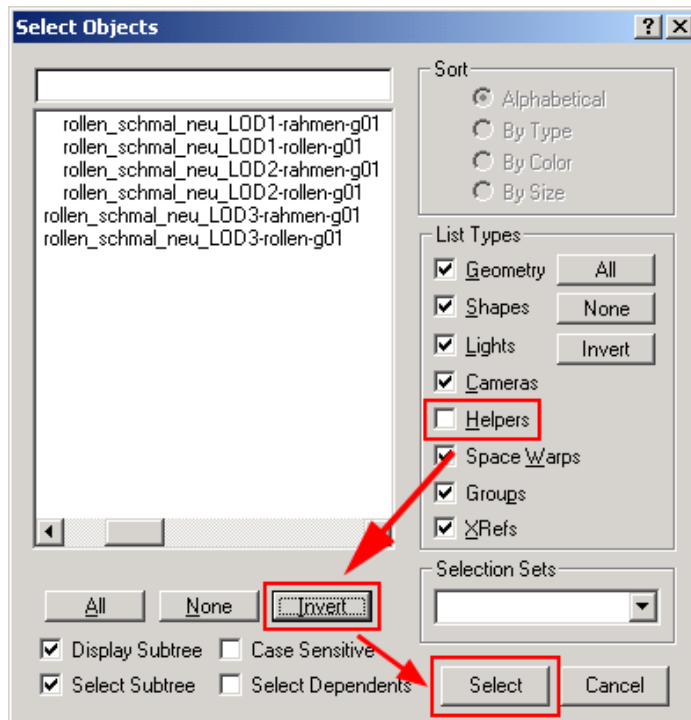


Bild 14: Selektionswerkzeug von 3D Studio Max: Auswahl von *Dummy*-Objekten

Nachdem die *Dummies* gelöscht sind, liegen die verbleibenden Modelle unorganisiert in der Szene vor. Wie in obigem Bild zu sehen ist (es handelt sich um ein Stück Transportstrecke direkt nach den Walztürmen), besteht ein Anlagenteil aus mehreren Modellen – bei komplexeren Anlagenteilen aus mehreren Dutzend Modellen.

Daher empfiehlt es sich für die Arbeit in *3D Studio Max*, alle Modelle eines Anlagenteils mit einem *Dummy* zu verknüpfen und so die einzelnen Modelle innerhalb der Szene zu organisieren.

Die Modelle eines Anlagenteils, die in der fertigen 3D-Szene nicht animiert werden, wurden zu einem Modell zusammen gefasst, um die Anzahl der einzelnen Modelle in *NeMo* klein zu halten und so den Überblick besser zu behalten. Diese Maßnahme wurde im Rahmen der im folgenden Abschnitt beschriebenen Reduzierung der Materialien durchgeführt, da die Bearbeitung und das Zusammenführen von identischen Materialien nach dem Zusammenfassen von Modellen auf-

Die Produktion

Eine Schritt-für-Schritt-Anleitung

wändiger ist. Die dafür nötige Auswahl der Optionen wird im folgenden Abschnitt beschrieben.

Für den Export in das nmo-Format werden dann nur die Modelle selbst ohne die *Dummys* ausgewählt.

2.2.2 Die Texturen und Materialien

NeMo beherrscht die drei gängigsten Schattierungsmodi Flat, Gouraud und Phong. Die Schattierungsmodi der Anlagenteil wurden unverändert übernommen.

Die Szene wird in der Interaktiven 3D-Visualisierung von einem Hauptlicht und einem Aufheller-Licht gleichmäßig erhellt.

Auch die Texturen müssen in ein von *NeMo* akzeptiertes Format überführt werden. Datenreduzierungspotenzial gibt es vor allem bei den Materialien. Im folgenden wird beschrieben, wie die Texturen in *NeMo* überführt wurden und wie die Zahl der verschiedenen Materialien klein gehalten wurde.



NeMo übernimmt Änderungen des UV-Abstands bei Texturen nicht. Die Texturen müssen also so gestaltet werden, dass sie ohne weitere Modifikationen auf Modelle gelegt werden können.

Überführung der Texturen

Die Texturen der Medit-Modelle liegen im SGI RGB-Format vor.

NeMo kann folgende Formate importieren:

- BMP
- TGA
- JPG
- DIB
- PCX
- TIFF

Ein Alphakanal für Transparenzen, wie er im TGA- oder TIFF-Format enthalten ist, wird leider nicht unterstützt. Um transparente Bereiche in einer Textur zu definieren, muss der Textur eine weitere Bilddatei mit Graustufen zugewiesen werden,

Die Produktion

Eine Schritt-für-Schritt-Anleitung

deren Grauwerte die Transparenz bestimmen. Ob Schwarz deckend und Weiß transparent ist oder umgekehrt, kann in NeMo bestimmt werden, indem ein Kontextmenü mit der rechten Maustaste auf die Alpha-Maske aufgerufen wird.

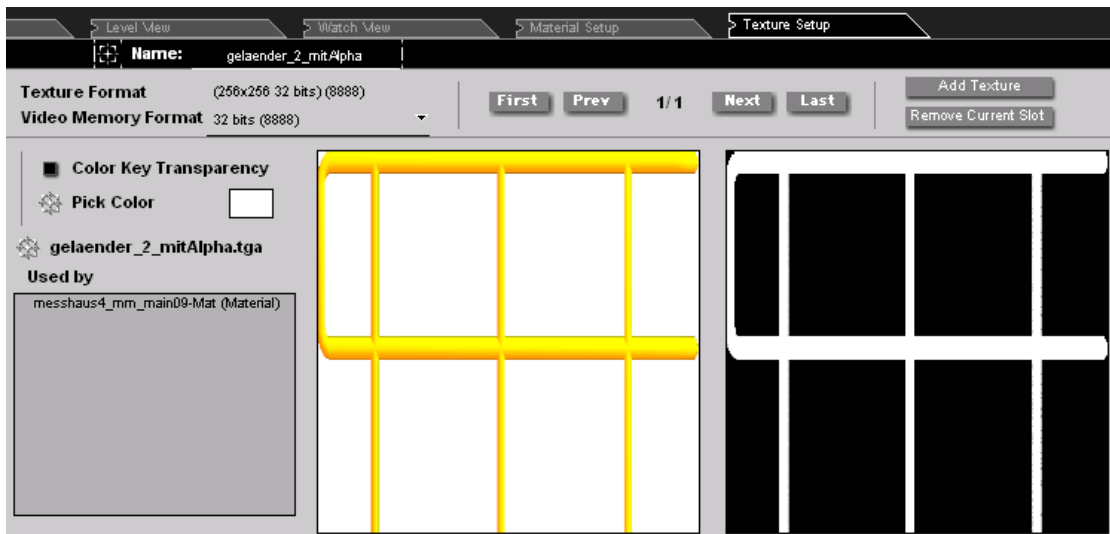


Bild 15: Textur-Setup und Alpha-Maske für das Geländer in NeMo

Typischerweise ist in der NeMo-Dokumentation kein Hinweis zu finden, was die bis zu 13 Mischoptionen für Textur und Alpha-Maske im Material-Setup bedeuten und welche davon zu wählen sind, um den gewünschten Transparenz-Effekt zu erzielen. Die Standardeinstellung beispielsweise ignoriert die Alpha-Maske.

Das folgende Bild zeigt die Einstellung, die durch Probieren das gewünschte Ergebnis für das Geländer erzielte:

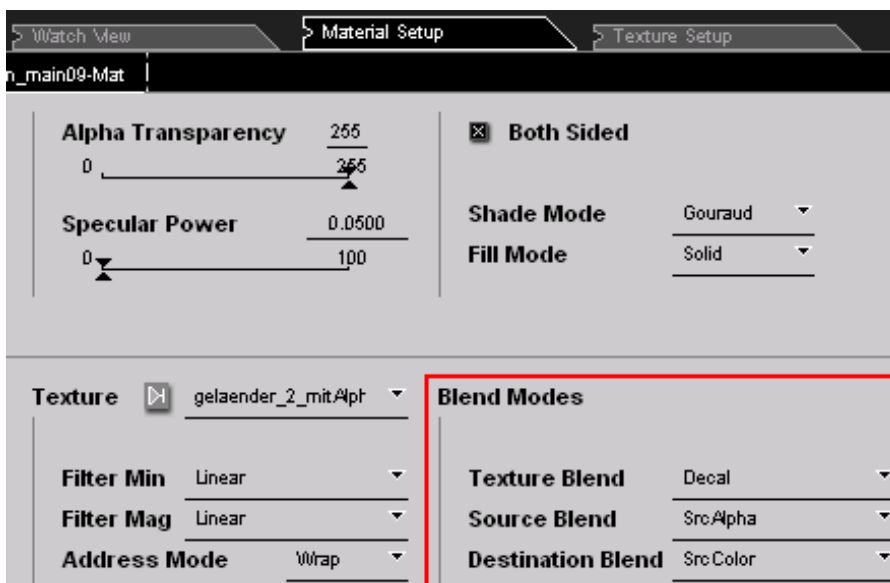


Bild 16: Material-Setup in NeMo

Die Produktion

Eine Schritt-für-Schritt-Anleitung

In der 3D-Ansicht von *NeMo* werden die deckenden Bereiche der Textur teilweise nicht richtig angezeigt, wenn im Blickfeld dahinter eine halbtransparentes Material ist. Das darf nicht verwirren – im Browser-PlugIn ist die Darstellung korrekt.

Die Texturen müssen OpenGL-konform sein, d. h. quadratisch, und ihre Maße müssen in Pixeln gemessen eine Potenz von 2 haben, also 1x1, 2x2, 4x4, 8x8, 16x16, 32x32, 64x64, 128x128 usw. Die vorliegenden Texturen der SVP sind nicht alle quadratisch.

Es gibt zwei Möglichkeiten, damit umzugehen:

1. Stauchen der Textur

Bei langgestreckten Texturen wie z. B. dem Geländer der Walztürme oder den Öfen führt das zwangsläufig zu Unschärfen, da diese Texturen auf ein quadratisches Format gestaucht werden müssen (und beim Rendern ja wieder in die ursprüngliche Länge gezogen werden entsprechend der Modellform). Dieser Nachteil wird aber in diesem Projekt dadurch realtiviert, dass der Einsatz z. B. der Geländertextur Hunderte von Polygonen in der Szene einspart. Aus Zeitgründen wurde diese einfachere Methode für die interaktive 3D-Visualisierung angewendet.¹⁸

2. Verwendung von Multi-/Unterobjekt-Materialien

NeMo unterstützt die Vergabe von mehreren Materialien (und damit Texturen) an ein Modell. Somit ist es möglich, eine langgestreckte Textur in quadratische Einheiten aufzuteilen, diese auf die gleichermaßen aufgeteilte Geometrie zu verteilen und die Geometrie anschließend wieder wie im folgenden Abschnitt beschrieben zusammenzuhängen. Diese Methode ist aufwändiger. Sie ist auf jeden Fall in *3D Studio Max* vorzunehmen. In *NeMo* sind solche Operationen nicht möglich.

Alle Texturen wurden mit Photoshop bearbeitet und konvertiert. Photoshop kann von Haus aus das RGB-Format nicht lesen. Es gibt aber verschiedene Webseiten, von denen ein entsprechendes Photoshop-PlugIn frei erhältlich ist, z. B. www.highend3d.com.

¹⁸ Laut Auskunft meines fachlichen Betreuers hat allerdings die Performance der 3D-Visualisierung der OpenGL-basierten SVP durch den Einsatz nicht OpenGL-konformer Texturen stark gelitten.

Die Produktion

Eine Schritt-für-Schritt-Anleitung

Da die VRML-Dateien im Text-Format vorliegen, kann ihr Quellcode mit einem Texteditor eingesehen und verändert werden. Die Texturpfade lassen sich so einfach an die Projektpfade angleichen (im einfachsten Fall liegen die Textur-Dateien im gleichen Verzeichnis wie die VRML-Dateien).

3D Studio Max kann das RGB-Format nicht verarbeiten, wohl aber das JPG- und TGA-Format. VRML hingegen unterstützt das RGB- und das JPG-, nicht aber das TGA-Format. Um die Texturen in *3D Studio Max* kontrollieren zu können, wurden in diesem Projekt RGB-Texturen ohne transparente Bereiche in JPG-Texturen und solche mit transparenten Bereichen in TGA-Texturen konvertiert. Die TGA-Texturen können dann auch in *NeMo* verwendet werden.

Vor dem Import der VRML-Modelle in *3D Studio Max* wurden im VRML-Quellcode die Dateiendungen der RGB-Texturen, die keine transparenten Bereiche enthalten, in .jpg geändert. Für den Import müssen die JPG-Dateien dann im durch den Pfad spezifizierten Verzeichnis liegen.

Beispiel:

```
texture DEF          caster_3-jpg_3          ImageTexture
          {url "caster_3.jpg"}
```

Die Dateiendungen von Texturen mit transparenten Bereichen müssen im VRML-Quellcode nicht verändert werden. Diese Texturen können im Material-Editor von *3D Studio Max* einfach durch TGA-Texturen ersetzt werden.

Somit sieht der eingeschlagene Überführungspfad für die Texturen wie folgt aus:

- RGB ➔ JPG
 - Ändern der Dateiendung im VRML-Quellcode von .rgb in .jpg
 - Ändern der Größe in quadratisches Format
 - Konvertieren der RGB-Datei in JPG-Datei, Bereitstellen in entsprechendem Verzeichnispfad
- RGB ➔ TGA
 - Ändern der Größe in OpenGL-konformes Format
 - Erstellen der Alpha-Maske
 - Konvertieren der RGB-Datei in TGA-Datei, Bereitstellen in entsprechendem Verzeichnispfad
 - Nach Import des Modells in *3D Studio Max* Ersetzen der RGB-Texturen durch TGA-Texturen in betreffenden Modellen. Die quadratischen Texturen werden dabei in das Format der ursprünglichen Texturen „gestreckt“.

Die Produktion

Eine Schritt-für-Schritt-Anleitung

NeMo bietet die Option, Texturen einer JPG-Kompression zu unterziehen. Ein Versuch mit der oben gezeigten Geländetextur und der Einstellung „Quality 100%“ erzielte das folgende sehr unbefriedigende Ergebnis:

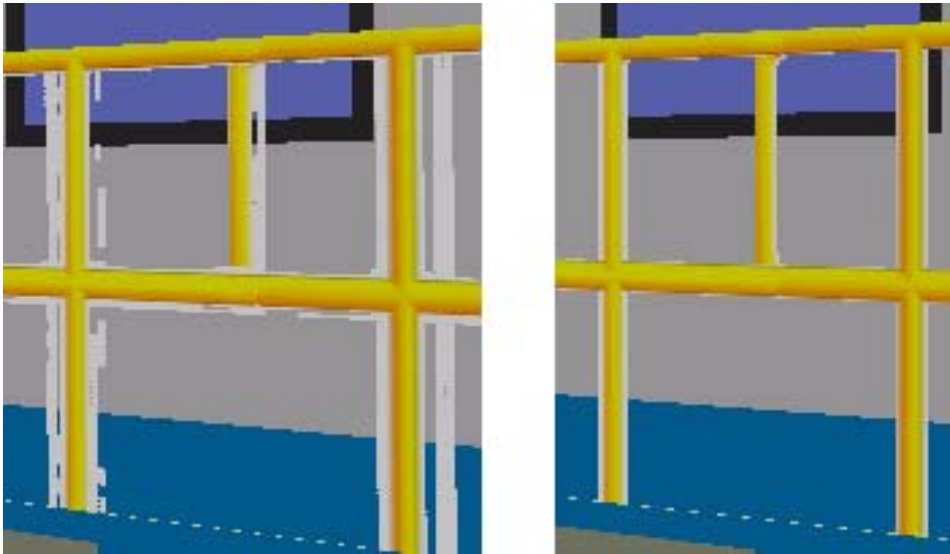


Bild 17: Links: Geländetextur in NeMo JPG-komprimiert, rechts: unkomprimiert

Reduzierung der Materialien

Beim Import-Pfad der Modelle von Medit über CosmoWorlds zu 3D Studio Max bekommt jedes Material eines Modells einen eindeutigen Namen. Das führt dazu, dass ein Material mit bestimmten Eigenschaften, z. B. das Blau vieler Stahlteile hundertfach in der Szene und dutzendfach in einem Modell vorkommt. Um den Verwaltungsaufwand für diese Materialien klein zu halten, wurden alle vorkommenden Materialien in *3D Studio Max* identifiziert und mit eindeutigen Namen versehen. Bei jedem weiteren in die Szene importierten Modell wurde geprüft, ob seine Materialien bereits vorhanden waren, und, wenn nicht vorhanden, mit Namen versehen und dem Material-Pool hinzugefügt. So wurde erreicht, dass jedes Material nur einmal in der Szene vorkommt und von mehreren Modellen genutzt wird.

Um auch die Anzahl der einzelnen Modelle innerhalb eines Anlagenteils zu reduzieren, wurden diese, soweit möglich und sinnvoll, mit dem Modifikator „Anhängen“ zu einem Modell zusammen gefasst. Jedes Modell, dem in der endgültigen Szene keine Animationen oder Skripte zugewiesen werden sollen, kann mit anderen Modellen des selben Anlagenteils sinnvoll zusammengefasst werden.

Die Produktion

Eine Schritt-für-Schritt-Anleitung



Dabei ist immer darauf zu achten, bei den Anhängoptionen "Material-IDs dem Material zuordnen" und „Material und IDs packen“ auszuwählen. So wird erreicht, dass identische Materialien zu einem Material zusammengefasst werden und nicht mehrfach vorkommen.

Im folgenden sind die einzelnen Schritte aufgeführt:

1. Modell auswählen, an das weitere Modelle angehängt werden sollen
2. Im „Ändern“-Modifikator „Anhängen“ auswählen
3. Im „Objekt auswählen“-Dialog anzuhängende(s) Modell(e) auswählen
4. In den Anhängoptionen „Material-IDs dem Material zuordnen“ und „Material und IDs packen“ auswählen

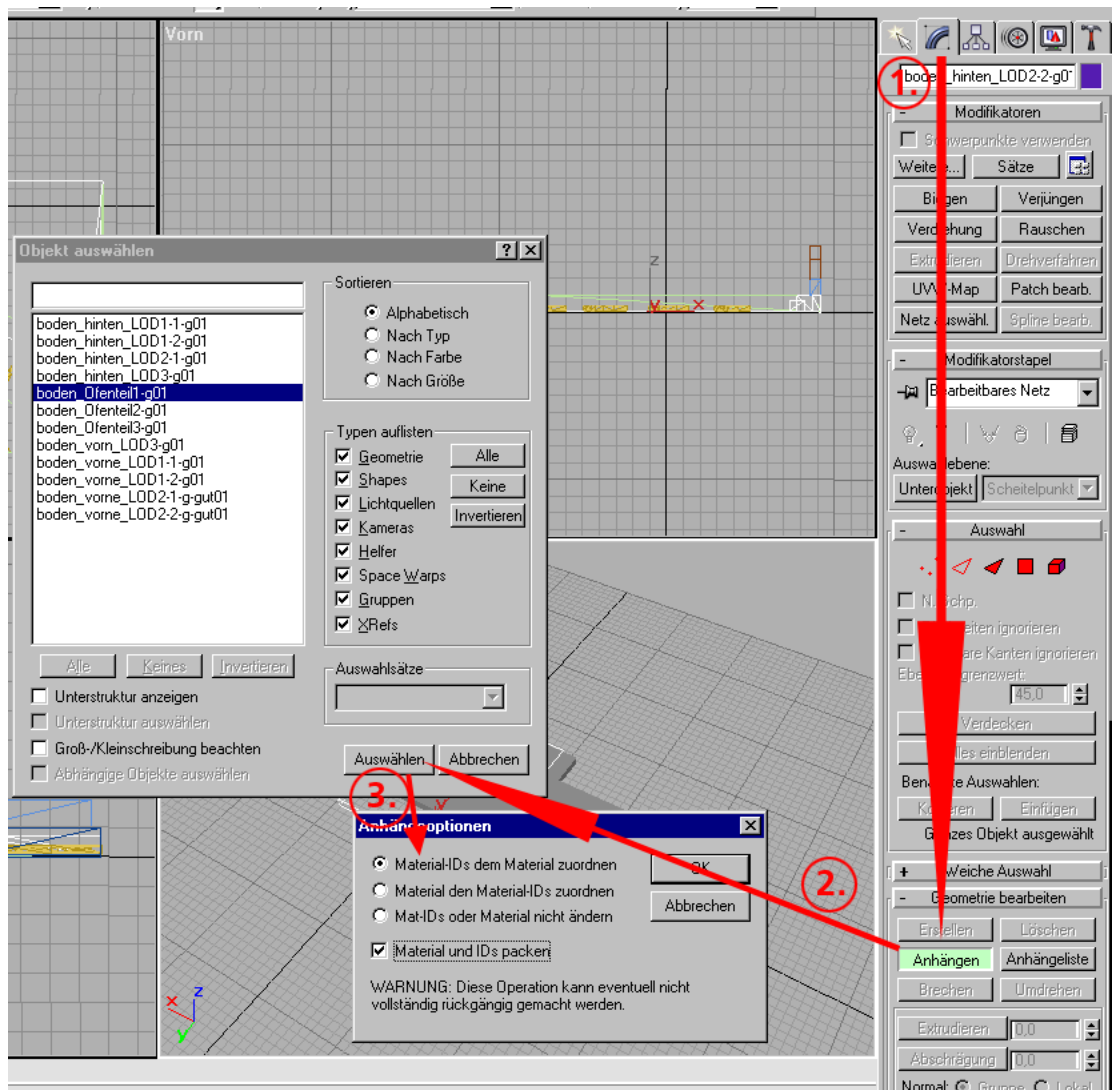


Bild 18: Zusammenfügen von Modellen und ihren Materialien in 3D Studio Max

Die Produktion

Eine Schritt-für-Schritt-Anleitung

Gemessen in kB konnte die Größe von *3D Studio Max*-Dateien einzelner Anlagenteile so um ca. 10 bis 30 Prozent reduziert werden. Folgendes Bild zeigt zum Vergleich links einen Auszug der Materialliste des Casters direkt nach dem VRML-Import, rechts die Materialliste des Casters nach der Zusammenfassung aller im Modell vorkommenden Materialien, wobei in diesem Modell noch alle LOD enthalten sind. Die Zahl der Modelle verringerte sich von 152 auf 10, die der einzelnen Materialien von 152 auf 34, die Dateigröße verringerte sich von 1.575 kB auf 1.483 kB. Beim Abschluss dieser Arbeit ist die *NeMo*-Szene 1.481 kB groß.

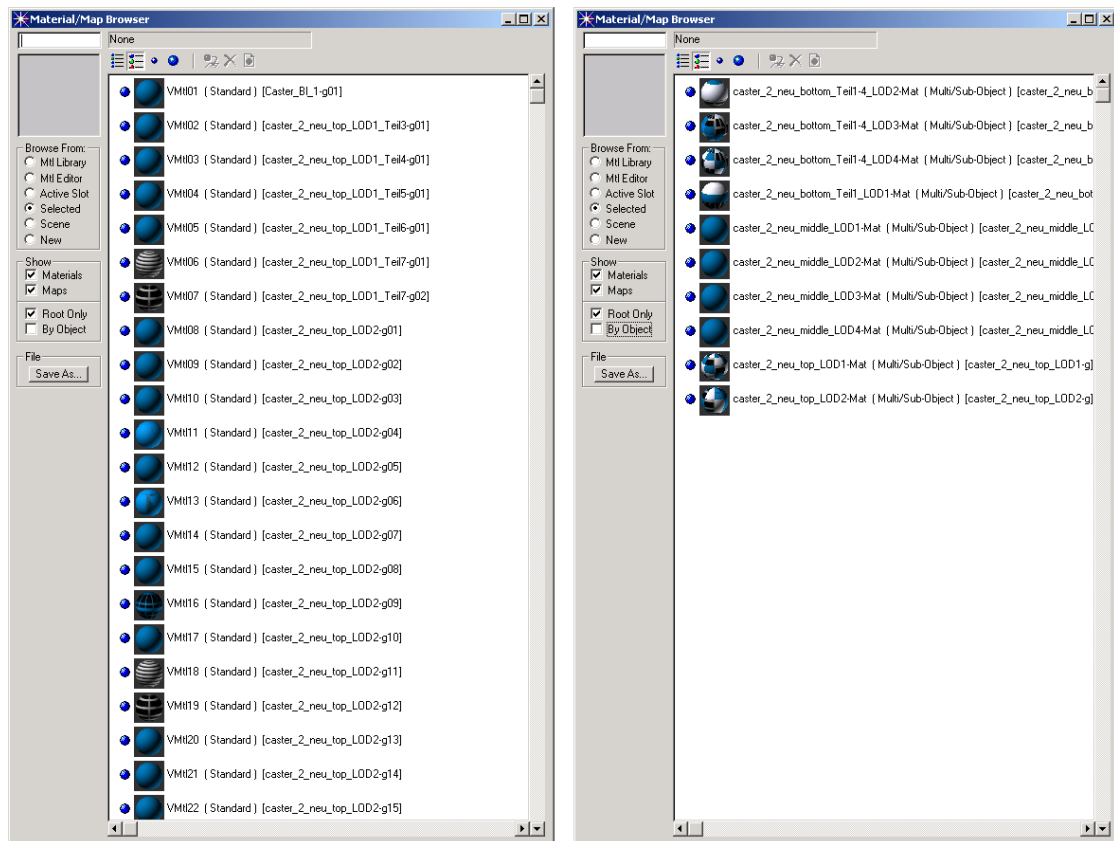


Bild 19: Links: Material-Liste des Casters nach dem VRML-Import, rechts: Material-Liste nach dem Zusammenfassen von Materialien und Modellen

Zusammenfassend kann festgestellt werden, dass die Materialien- und Polygon-Reduzierung eine sehr zeitintensive Maßnahme ist. Für Projekte in der Größenordnung wie der SVP muss frühzeitig geplant werden, ob und in welchem Umfang eine Visualisierung wie die hier produzierte vorzunehmen ist. Wenn ja, dann empfiehlt es sich, rechtzeitig die entsprechenden Maßnahmen zu treffen, also z. B. Namenskonventionen für Modelle zu vereinbaren und einzuhalten oder rechtzeitig die Modelle im adäquaten Format für das 3D-Autorenwerkzeug bereit zu stellen.

2.2.3 Die Animationen

Vor Beginn der Visualisierung gibt der Benutzer folgende Informationen an, die die Visualisierung beeinflussen:

- Anzahl der zu gießenden Brammen, jeweils beide Ofenstränge
- Länge jeder Bramme
- Auftragsreihenfolge, in der die Brammen der Walzanlage zugeführt werden

Alle Animationen müssen nun so koordiniert werden, dass in der 3D-Visualisierung die Brammen von den Schwenkfähren in der gewünschten Reihenfolge der Walzanlage zugeführt werden. Dieses komplexe und dynamische Zusammenspiel von Animationen ist das Kernstück der Interaktiven 3D-Visualisierung.

Die Animation der Brammen und der Schwenkfähre sowie einige einfachere Animationen von beweglichen Teilen der Gieß-Walz-Anlage erfolgt durch Verwendung der entsprechenden *Building Blocks* in den grafischen Skripten in *NeMo*.

Bevor im einzelnen erläutert wird, wie das Zusammenspiel aller beweglichen Teile erreicht wurde, werden die grundsätzlichen Möglichkeiten aufgezeigt, die *NeMo* für Animationen bietet.

Die Interaktive 3D-Visualisierung enthält ausschließlich Bewegungsanimationen, also keine Animationen von z. B. Materialien oder Texturen. *NeMo* bietet drei Möglichkeiten, um Bewegungsanimationen zu erstellen:

- *Pfadanimationen*
Das Objekt wird entlang einer 3D-Kurve bewegt. Die 3D-Kurve kann in einem 3D-Programm erstellt und importiert oder in *NeMo* selbst erzeugt werden (bequemer ist i. d. R. jedoch die Erstellung in einem 3D-Programm).
- *Explizite Transformationen mittels Building Blocks*
NeMo besitzt verschiedene *Building Blocks* für direkte Transformationen wie Rotation, Skalierung und Translation sowie für indirekte Transformationen. Das können z. B. Funktionen sein, die ein Objekt auf ein anderes Objekt ausrichten oder die Y-Position eines Charakters an einem definierten Boden-Objekt ausrichten.
- *Animationen, die komplexe Bewegungen enthalten*, können in einem 3D-Programm erstellt und anschließend in *NeMo* importiert werden. Zur Steuerung solcher Animationen gibt es einen

Die Produktion

Eine Schritt-für-Schritt-Anleitung

speziellen Satz von Building Blocks (vgl. Anlage A). Das betrifft im wesentlichen Charaktere, die z. B. Gliedmaßen bewegen müssen.

Für die Interaktive 3D-Visualisierung wurden aus zwei Gründen ausschließlich explizite Transformationen eingesetzt:

- Es handelt sich um einfache geradlinige („von Punkt A nach Punkt B“) oder lineare Bewegungen (z. B. Rotation der Drehtürme, Skalieren der Brammen).
- Die dafür zur Verfügung stehenden *Building Blocks* bzw. ihre *Parameter* erlauben es, z. B. die Zielkoordinaten einer Translation während der Laufzeit präzise zu variieren. Das ist von enormer Bedeutung z. B. bei den Bewegungen der Brammen rund um den Fahrenbereich. Da die Länge der Brammen vom Benutzer bestimmt wird, sind die Zielkoordinaten jeder Brame nicht vorher-sagbar.

Bei den anderen zwei der o. g. Animationsarten könnte zwar z. B. das Ende einer linearen Translation wie die Fortbewegung einer Brame durch den Ofen mit einer Prozentangabe im *Building Block Set Animation Step on 3D Entity* während der Laufzeit bestimmt werden. Diese Möglichkeit bietet aber nicht die gleiche Präzision wie z. B. die Änderung der X-Koordinate des *Translate Building Blocks*.

Im folgenden wird das Prinzip und die Funktionsweise der *Building Blocks* erklärt. (*Building Block* wird hier durch BB abgekürzt, um den Lesefluss nicht zu stören.)

Jeder BB besitzt links einen *Input*, bei dessen Aktivierung der BB ausgeführt wird, und rechts einen *Output* zur Aktivierung nachfolgender BBs.

Je nach Funktion besitzt ein BB einen *Input*, über den er aktiviert wird und einen *Output* zur Aktivierung des nächsten BB sowie optional oben eingehende *Input Parameter*, unten ausgehende *Output Parameter* und weitere optionale *Inputs* und *Outputs*. Optionale *Inputs* und *Outputs* dienen z. B. zum Durchlaufen von Schleifen bei iterativ arbeitenden BBs oder beispielsweise zum Beenden der Aktivität eines *Wait Message* BBs. *Input Parameter* sind das NeMo-Äquivalent zu Funktionsparametern in traditionellen Programmiersprachen, *Output Parameter* stellen Rückgabewerte von Funktionen dar. *Input Parameter* können direkt in das grafische Symbol für *Output Parameter* geleitet werden. Sie können auch Kopien von *Output Parametern* anderer BBs sein, die im gleichen oder in anderen Skripten der Szene existieren und heißen dann *Shortcuts* (das entspricht z. B. den Zeigern in einer Programmiersprache wie C).

Die Produktion

Eine Schritt-für-Schritt-Anleitung



Bild 20: Building Block in NeMo

Die meisten BBs wirken sich ohne Angabe eines *Target Parameters* auf das Objekt aus, in dessen Skript sie verwendet werden (andere BBs arbeiten objektunabhängig, z. B. die BBs der Gruppe *Calculator*). Bei Aktivierung und Belegung des *Target Parameters* mit einem Wert kann ein BB auf ein fremdes Objekt angewendet werden. So wird zum Beispiel das Öffnen und Schließen der Klappen des Ofenausgangs gesteuert, wenn er von den Brammen passiert wird.

Die sequenzielle Verbindung zweier BBs hat als Eigenschaft das sog. *frame delay*, mit dem festgelegt werden kann, wie viele Renderperioden bis zur Aktivierung des nachfolgenden BBs verstreichen sollen. So können z. B. Animationen verlangsamt werden, allerdings werden diese dann ruckelig.



Generell ist es wünschenswert, möglichst viele BBs innerhalb einer Renderperiode ausführen zu lassen. In einer Schleife, die von einer Sequenz von BBs gebildet wird, wenn einer der Iterations-BBs benutzt wird, ist jedoch immer ein *frame delay* von mind. 1 erforderlich, da sonst die gesamte Iteration innerhalb einer Renderperiode durchgeführt wird.



Es hat sich gezeigt, dass bei Zähliterationen, wie z. B. der *Linear Progression* BB, nicht zu Ende gezählt wird, wenn für die Interaktive 3D-Visualisierung ein hoher Zeitfaktor eingestellt ist, weil dann weniger Iterationsschritte zur Verfügung stehen. Das hat zur Folge, dass z. B. ein Drehturm seine 180°-Drehung nicht ganz zu Ende führt. Daher ist es erforderlich, direkt nach der Iteration eine absolute Positionierung oder Ausrichtung mit den gewünschten Werten vorzunehmen.

Skripte stellen Verhalten von Objekten dar. Ein Skript kann jedem Objekt in der Szene zugewiesen werden, auch einer *Scene* bzw. bei Spielen einem *Level* selbst zur globalen Kontrolle. Ein Skript enthält eine beliebige Sequenz von BBs. BBs können auch parallel angeordnet werden, sie werden dann tatsächlich parallel ausgeführt. Das folgende Bild zeigt exemplarisch den ersten Teil des Animations-Skripts einer Bramme. Die Bramme wird hierbei auf ihre gewünschte Länge skaliert, wodurch der Eindruck entsteht, dass sie von der Schneidemaschine ausgegeben wird.

Die Produktion

Eine Schritt-für-Schritt-Anleitung

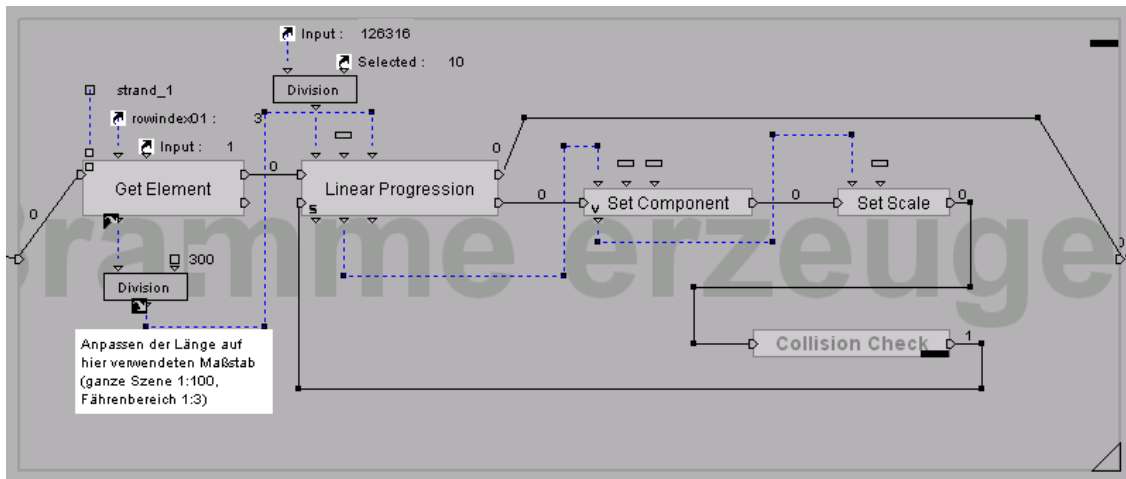


Bild 21: NeMo-Building Block „Bramme erzeugen“ zur Skalierung einer Bramme auf die gewünschte Länge¹⁹

Der *Get Element* BB liest aus der Brammen-Tabelle den Längenwert für eine spezifische Bramme und übergibt ihn an eine *Parameter Operation*, die den Wert mittels Division auf einen kleineren Maßstab herunter rechnet.²⁰ Das Ergebnis wird an den *Linear Progression* BB übermittelt, der in einer Schleife die Skalierungsschritte durchführt. Die Zeitdauer dafür wurde in der Ereignis-Tabelle²¹ der SVP ermittelt. Der Zeitwert wird hier in einer weiteren *Parameter Operation* durch den eingestellten Zeit-Faktor dividiert, der in diesem Beispiel den Wert 10 hat. Der *Loop Out*-Ausgang von *Linear Progression* ist mit dem *Set Component* BB verbunden, der die Skalierungswerte für die Y- und Z-Achse auf 1 feststellt, während der X-Skalierungswert bei jedem Iterations-Durchlauf erhöht wird. Der *Output Parameter* von *Set Component* ist ein Vektor mit den Werten für X, Y und Z, der an den *Set Scale* BB übergeben wird, welcher wiederum das Brammenmodell auf den aktuellen Wert skaliert.

Der *Collision Check* BB enthält ein Skript, das dazu dient, diese Animation zu pausieren, falls eine Kollision mit der Vorgängerbramme auftritt.²² Dieser BB ist

¹⁹ Hier wird ein prinzipieller Nachteil von grafischen Skripte sichtbar: Hat die Autorenumgebung keine Funktionen zur dokumentationsmäßigen Ausgabe der Skripte, so gestaltet sich eine Dokumentation recht aufwändig, da die Skripte nur per Screenshot in Dokumente eingebunden werden können. Außerdem werden komplexe Skripte schnell unübersichtlich. Auch eine Suche nach bestimmten Kommentaren oder Funktionen, wie sie die einfachsten Texteditoren bieten, ist nicht möglich, wenn das Autorenwerkzeug die dafür notwendige Funktionalität nicht bereitstellt. Bei NeMo ist dies leider nicht der Fall.

²⁰ Der gesamte Ofenbereich ist, wie auch in der SVP, auf der X-Achse auf den Maßstab 1:3 skaliert, um die 3D-Szene von der Gesamtansicht „handlicher“ zu machen.

²¹ vgl. Abschnitt 2.2.4

²² Tatsächlich dürfte eine neue Bramme erst gegossen werden, nachdem die Vorgängerbramme diesen Be-

Die Produktion

Eine Schritt-für-Schritt-Anleitung

ein Beispiel für die Möglichkeit, Skripte wieder zu verwenden. Der *Output* des *Collision Check* BB führt in den *Loop*-Input des *Linear Progression* BB, womit der nächste Iterationsschritt eingeleitet wird. Am Ende der Progression wird der *Out*-Ausgang aktiviert, der wiederum den Ausgang des „Bramme erzeugen“-Skripts aktiviert, das selbst als wiederverwendbares BB fungiert.

Auf die oben beschriebene Art und Weise funktionieren alle Animationen, die dem Transport der Brammen durch die Anlagenteile dienen.



Es gibt übrigens einige unschöne Eigenheiten, die den 3D-Autor bei seinen Routearbeiten verärgern: So enthält z. B. der *Switch On Parameter* BB einen unnötigen Dialog bei der Einstellung jedes weiteren Parameters. Der Benutzer wird in einem Dialogfenster aufgefordert, den Typ des neuen Parameters anzugeben, obwohl nur Parameter gleichen Typs zulässig und sinnvoll sind. Tatsächlich wird dem Parameter von NeMo auch der zulässige Typ zugewiesen - unabhängig davon, welche Auswahl der Benutzer in diesem Dialog trifft.



Das Konzept der wiederverwendbaren BBs in NeMo legt den Vergleich zu Funktionen in traditionellen Programmiersprachen nahe. Der Vergleich hinkt jedoch: Jeder wiederverwendete BB erzeugt weiteren Code und vergrößert die 3D-Szene.



Der Vorteil der wiederverwendbaren BBs wird sehr stark relativiert durch die Tatsache, dass *Shortcuts* zu Parametern in anderen Skripten verloren gehen, wenn ein Skript auf Datenträger abgespeichert wird. Somit ergibt sich nach dem Laden eines Skriptes erhebliche und fehlerbehaftete Mehrarbeit für die Vervollständigung des Skriptes. Eine Bramme besitzt beispielsweise aufwändige Skripte, die ihren gesamten „Lebenslauf“ abbilden. Darin sind einige Parameter mit fixen Werten als auch *Shortcuts* zu skriptfremden Parametern enthalten, z. B. der Zeitfaktor.



Nach dem Laden der Projektdatei kann es vereinzelt vorkommen, dass *Shortcuts* ihre Verbindung zu *Input Parametern* oder ein BB seine Verbindung zum nächsten BB verliert. Wenn ein Skript nicht wie sonst üblich funktioniert, kann dies die Ursache dafür sein.



Eine ähnliche Auswirkung kann die nachträgliche Erweiterung oder Reduzierung einer Tabelle um Spalten haben. Auch wenn zu keinem Zeitpunkt auf die Werte dieser Spalten zugegriffen wurde, muss trotzdem bei allen BBs, die auf diese Ta-

reich verlassen hat. Der Verzicht auf die dafür notwendige Prüfung, die von der Länge der Brammen abhängig ist, ist ein Eingeständnis an die starke Vereinfachung.

Die Produktion

Eine Schritt-für-Schritt-Anleitung

belle zugreifen, einmal der *Edit Parameters*-Dialog geöffnet werden, damit dieser NeMo-intern bezgl. der Anzahl der Spalten aktualisiert wird. Andernfalls wird dieser BB zur Laufzeit nicht ausgeführt. Diese umständliche Verfahrensweise betrifft alle BBs, die auf ein Objekt angewendet werden, dessen Eigenschaften nachträglich geändert werden und. Das ist insbesondere beim Testen der 3D-Szene im Browser ärgerlich, weil dabei keine Möglichkeiten der Skript-Verfolgung zur Verfügung stehen. Während der Projektarbeit kam es vereinzelt vor, dass das gesamte Browserfenster nicht mehr reagierte, weil ein BB zur Ausführung kam, dessen Parameter nicht mit einem Wert belegt war. In der Autorenumgebung hingegen zeigte diese Nachlässigkeit zufälligerweise keine Auswirkung, weil dieser BB beim Testen gerade nicht zur Ausführung kam.

2.2.4 Die Interaktivität

In der SVP wird jede einzelne Bewegung von Ereignissen angestoßen und gesteuert, die von den Prozessreglern erzeugt werden. Die Interaktive 3D-Visualisierung interagiert nicht mit simulierten Prozessreglern. Sie ist aber so ausgelegt, dass in einer Ereignis-Tabelle animationsrelevante Ereignisse aus einem echten Simulationslauf mit ihren *timesteps* abgelegt werden können. Diese Ereignisse steuern dann den Visualisierungsablauf, indem sie vorgeben, wann bestimmte Objekte bestimmte Animationen durchführen, z. B. die Erzeugung der 3. Bramme auf Strang 1. Damit können z. B. beim Kunden verschiedene Simulationen präsentiert werden.

In der SVP steuern 22 verschiedene Ereignisse die Visualisierung (vgl. Anlage C). Die Interaktive 3D-Visualisierung verwendet das folgende vereinfachte Ereignismodell:

Ereignisnummer	Ereignisbeschreibung
Ereignis 11 bzw. 12	Drehung Drehturm Strang 1 bzw. 2
Ereignis 31 bzw. 32	Füllbeginn Tundish ²³ Strang 1 bzw. 2
Ereignis 101 bzw. 102	Gießbeginn Caster Strang 1 bzw. 2
Ereignis 121 bzw. 122	Erzeugen einer Bramme
Ereignis 501 bzw. 502	Öffnen Ofeneintritt Strang 1 bzw. 2

Tabelle 2: Ereignisse in der Interaktiven 3D-Visualisierung

²³ Der Tundish ist ein Auffangbehälter für den flüssigen Stahl direkt unterhalb der Gießöffnung eines Drehturms.

Die Produktion

Eine Schritt-für-Schritt-Anleitung

Da die Benutzereingaben anstelle von simulierten Prozessreglern die nachfolgenden Animationen beeinflussen, sind weitere Ereignisse nicht notwendig. Die auf die Erstellung einer Bramme folgenden Animationen werden im weiteren durch Skripte und Informationsaustausch zwischen Skripten gesteuert.

Die Konfiguration der Interaktiven 3D-Visualisierung für verschiedene Präsentationen wird so auch vereinfacht.

Für diese Arbeit wurden die Ereignisse und ihre *timestamps* eines Simulationslaufs mit jeweils 10 Brammen auf jedem Strang verwendet.

Im folgenden werden die zwei zentralen Skripte beschrieben, die alle anderen Animationsskripte in der Szene steuern, der „Event Manager“ und der „ferry manager“ (vgl. Anlagen D und E).

Das Skript „Event Manager“ aktiviert zu bestimmten Zeitpunkten bestimmte Animationsskripte. Nach dem Start der Visualisierung wird der „Event Manager“ ausgeführt. Ein *Time Bezier Interpolator* BB durchläuft eine 70.000.000 ms dauernde Schleife, in der die abgelaufene Zeit oben rechts im 3D-Bild ausgegeben wird. Gleichzeitig wird in einer anderen BB-Sequenz im gleichen Skript das erste Ereignis aus der Ereignis-Tabelle gelesen. Der *timestamp* des Ereignisses wird mit der aktuellen Zeit verglichen. Bei Übereinstimmung oder wenn die Zeit des Ereignisses bereits verstrichen ist, wird die *Message* „do-something“ abgeschickt. Mit dieser *Message* ist die Ereignis-Nummer verknüpft. Von den Objekten, die auf den Erhalt von *Messages* warten, wird nur dasjenige sein Animationsskript ausführen, das die passende Ereignis-Nummer enthält. Nach dem Versenden der *Message* erfolgt die nächste Iteration im „Event Manager“.

Das Skript „ferry manager“ steuert die Animation der Brammen im Bereich der Schwenkfähren und die Schwenkfähren selbst. So werden die Brammen in der gewünschten Auftragsreihenfolge der Walzanlage zugeführt. Die Tabelle „ferry queue“ enthält die Namen der die Brammen repräsentierenden 3D-Objekte in der Auftragsreihenfolge²⁴. Beginnend bei der ersten Bramme wartet der „ferry manager“ darauf, dass sich Brammen bei ihm „anmelden“, indem sie sich (genau: das sie darstellende 3D-Objekt) in die Tabelle „requesting slabs“, sozusagen die Warteliste, eintragen. Die Bramme an der aktuellen Position in der „ferry

²⁴ Der erforderliche Parameter zur Änderung der Startreihenfolge aufgrund von Benutzereingaben wird der 3D-Szene über die JavaScript-Schnittstelle übermittelt. Näheres dazu am Ende dieses Abschnitts.

Die Produktion

Eine Schritt-für-Schritt-Anleitung

queue“ wird fortlaufend verglichen mit den Brammen in der Tabelle „requesting slabs“. Ist die richtige Bramme darunter, prüft der „ferry manager“ zunächst, ob die Fähre evtl. noch in Benutzung ist und schickt nach Erkennung des Freizeichens der Fähre die *Message* „faehre_frei“ an die wartende Bramme. Erst nach einer Bestätigung der *Message* durch die Bramme in Form einer Statusänderung liest der „ferry manager“ die nächste Bramme aus der „ferry queue“-Tabelle ein und vermeidet so eine Überschneidung von evtl. möglichen gleichzeitigen Statusänderungen.



Es hat sich leider gezeigt, dass *Messages* „verloren“ gehen können. Dies tritt insbesondere bei einem großen Zeitfaktor, wie z. B. 100, auf.²⁵ Daher sendet z. B. der „ferry manager“ fortlaufend die „faehre_frei“-*Message*, bis die Bestätigung durch die wartende Bramme erfolgt. Bei einer ersten Version des „ferry managers“, der anstatt der „requesting slabs“-Tabelle nur mit *Messages* arbeitet, trat häufig ein Verschränkungseffekt auf: Die Bramme, die tatsächlich laut Auftragsreihenfolge an der Reihe gewesen wäre, der Walzanlage zugeführt zu werden, durchlief fortlaufend eine Schleife, in der sie eine „Ich bin Bramme Nr. xx und will durch“-*Message* an den „ferry manager“ schickte und auf die *Message* mit dem Freizeichen wartete, während gleichzeitig der „ferry manager“ fortlaufend an genau diese Bramme das Freizeichen schickte und seinerseits auf die Empfangsbestätigung der Bramme wartete. Der Grund für diese Verschränkung liegt vermutlich darin, dass zufällig in der Renderperiode, in der z. B. der „ferry manager“ das Freizeichen an die Bramme schickte, im Skript dieser Bramme der *Wait Message* BB wegen anderer Rechenoperationen nicht aktiv war. Auch eine Änderung von *frame delays* brachte keine Abhilfe, sondern verschob den Zeitpunkt der Verschränkung nur.

Die entsprechende Bramme aktiviert dann die erforderlichen Skripte zur Rotation der Schwenkfähren und zu ihrer eigenen Rotation und Fortbewegung über die Schwenkfähren. Gleichzeitig aktiviert sie den *State* „ferry_in_use“, der den „ferry manager“ daran hindert, der nächsten wartenden Bramme die „faehre_frei“-*Message* zu schicken, und löscht ihren Eintrag in der Tabelle „requesting slabs“. Erst wenn die Bramme den Schwenkfährenbereich vollständig verlassen hat und in den Walzbereich weiter fährt, deaktiviert sie den *State* „ferry_in_use“.

²⁵ Die dieser Arbeit beigelegte Version der Interaktiven 3D-Visualisierung enthält den Zeitfaktor 100, um das beschriebene Problem zu demonstrieren. Die Online-Version wird einen kleineren Zeitfaktor enthalten.

Die Produktion

Eine Schritt-für-Schritt-Anleitung

Ergänzend wird hier noch das Skript erläutert, das für die Einstellung des Zeitfaktors in der 3D-Szene zuständig ist. Da dieses Skript Eingaben des Benutzers über die Benutzeroberfläche erwartet, wird anschließend die JavaScript-Schnittstelle des NeMo-ActiveX-Elements erläutert.

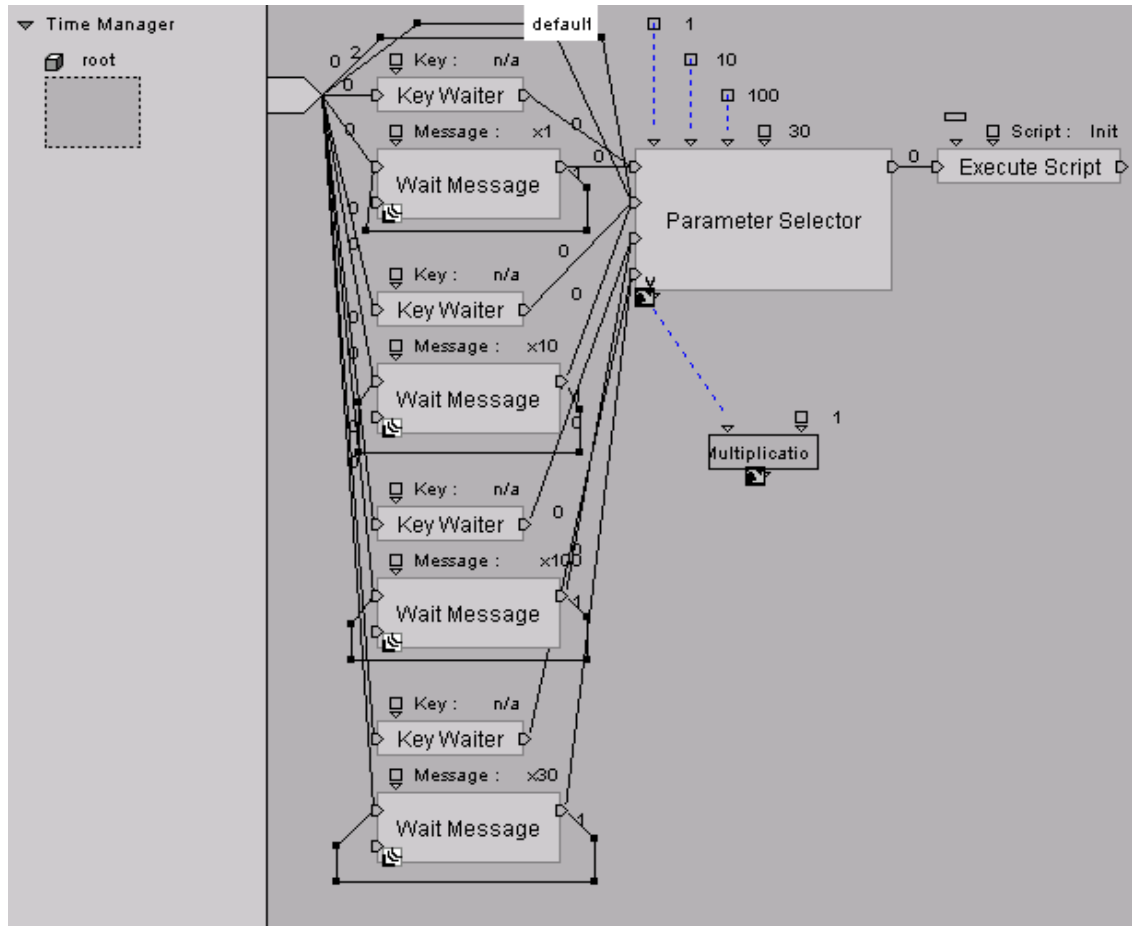


Bild 22: NeMo-Skript „Time Manager“ zur Einstellung des Zeitfaktors

Der *Key Waiter* wartet auf die Eingabe eines bestimmten Zeichens über die Tastatur. Das Zeichen ist in dem Parameter *Key* spezifiziert, der von oben in den *Key Waiter* eingeht. Wird die entsprechende Taste gedrückt, so wird der nächste mit dem *Key Waiter* verbundene BB aktiviert, in diesem Fall der *Parameter Selector*.²⁶

Der *Wait Message* BB wartet u. a. auf vordefinierte Mauseaktionen wie Klick und Doppelklick. Es lassen sich aber auch individuell definierte Messages als Parame-

²⁶ In der NeMo-Autorenumgebung gibt es keine Entsprechung für die JavaScript-Schnittstelle. Daher müssen während der Autorenenarbeit Mausklicks auf der Webseite durch Tastatureingaben ersetzt werden. Komplexe Funktionen wie beispielsweise die Eingaben der Brammenlängen können im Gesamtzusammenhang überhaupt erst im Zusammenspiel der 3D-Szene mit dem Browser entwickelt und getestet werden.

Die Produktion

Eine Schritt-für-Schritt-Anleitung

ter angeben. In diesem Fall sind es Zeichenketten, die per JavaScript an das NeMo-ActiveX-Element übergeben werden. Dies ist die einzige Möglichkeit, Informationen von der Webseite an die 3D-Szene zu übergeben!



Hier liegt ein entscheidender Nachteil von *NeMo Creation 1.0.1*. Die als *Message* in die 3D-Szene eingehenden Informationen können nicht als Parameter z. B. für die Längenbestimmung der Brammen verwendet werden, weil eine *Message* vom Datentyp *Message* ist und *NeMo* keine Funktionen zur Konvertierung von *Messages* in andere Datentypen anbietet. (Im Benutzerforum *swapmeet* wurde darüber großer Unmut geäußert. Ob Datentyp-Konvertierungen in *Dev 2.0* nun möglich sind, ist nicht bekannt, zumindest aber sind weitergehende Funktionen zur Zeichenkettenbearbeitung und eine erweiterte JavaScript-Schnittstelle in *Dev 2.0* enthalten.) Deshalb ist für jede in der Benutzeroberfläche mögliche Längeneingabe für jede Bramme ein *Wait Message* BB notwendig, der den entsprechenden Zahlenwert für ihre Länge in die Brammentabelle einordnet. Das macht für jede der 20 Brammen ein Skript erforderlich, das mit jeweils zehn *Wait Message* BBs die zehn möglichen Längeneingaben erwartet. Insgesamt 200 *Wait Message* und 200 *Set Elements* BB schaffen so eine Funktionalität, die in traditionellen Programmiersprachen mit einer einzigen Funktion abgedeckt werden könnte! Aus diesem Grund ist es verständlich, dass auf die Eingabe beliebiger Zahlenwerte zwischen 5.000 und 50.000, wie es in der SVP möglich ist, verzichtet wurde.

Der *Parameter Selector* weist nach Aktivierung durch den *Input n* dem *Output Parameter* den Wert des *Input Parameters n*, also den ausgewählten Zeitfaktor, zu.

Der dem *Parameter Selector* folgende *Execute Script* BB führt ein Skript aus, das die *timestamps* in der Ereignis-Tabelle dem Zeitfaktor entsprechend skaliert.



Wird ein Skript mit dem *Execute Script* BB aufgerufen, so beginnt die Ausführung des aufgerufenen Skripts einen Frame, also eine Renderperiode, später. Sollen in dem aufrufenden Skript nachfolgende Animationen synchron zu den Animationen im aufgerufenen Skript ablaufen, so ist davor ein *frame delay* von 1 einzustellen.

Das *NeMo-ActiveX-Element* wird mit folgendem Code in den HTML-Quelltext eingebunden:

```
<object classid="CLSID:C4925E65-7A1E-11D2-8BB4-00A0C9CC72C3" id="Virtools"
width="480" height="360" code-
base="http://player.virtools.com/downloads/player/Install2.0/Installer.exe#Ver-
sion=2,0,0,33" name="NeMoAX1">
  <param name=SRC value="../../../cmo/vneue_sms-welt.cmo">
```

Die Produktion

Eine Schritt-für-Schritt-Anleitung

```
<embed SRC="../../cmo/vneue_sms-welt.cmo" type="application/x-virttools"
pluginspage="http://player.virttools.com/" width="480" height="360" back-
color="&HFFFFFF" name="NeMoAX1">
</embed>
</OBJECT>
```

Dabei ist `NeMoAX1` der Name des ActiveX-Elements, mit dem es im weiteren referenziert wird. Mit `width=` und `height=` werden die Maße des 3D-Bildes in Bildschirmpunkten angegeben.

Der JavaScript-Befehl zur Übergabe einer Zeichenkette an das ActiveX-Element lautet:

```
document.NeMoAX1.SendMessageToObject(ObjectID,MessageName);
```

`ObjectID` muss dabei vorher ermittelt werden mit:

```
var ObjectID = document.NeMoAX1.GetObjectID(ObjectName);
```



Vom 3D-Autor wird hier erhöhte Aufmerksamkeit verlangt: Objektnamen dürfen in *NeMo* mehrfach vorkommen! Da die `ObjectID` jedoch über den Objektnamen ermittelt werden muss, ist es unbedingt notwendig, die betreffenden Objekte mit eindeutigen Namen zu versehen.

Für die Übermittlung der Auftragsreihenfolge an die 3D-Szene wurde folgendes Verfahren entwickelt, das eine Unmenge von *Wait Message* BBs zum Abfangen aller möglichen Kombinationen von Bramme-Position-Zuordnungen vermeidet: Eine Standard-Reihenfolge nach dem Reißverschlussprinzip wird zu Beginn festgelegt und in der Tabelle „ferry queue“ festgehalten (Bramme 1, Bramme 11, Bramme 2, Bramme 12, Bramme 3, usw.), wenn der Benutzer auf Strang 1 mehr als oder gleich viele Brammen wie auf Strang 2 wählt. Diese Reißverschluss-Reihenfolge wird umgedreht (also Bramme 11, Bramme 1, Bramme 12, Bramme 2, Bramme 13 usw.), wenn auf Strang 2 mehr Brammen als auf Strang 1 gewählt werden.

Bei jeder Tauschoperation, die der Benutzer macht, wird lediglich der entsprechende Positionsindex als *Message* an die 3D-Szene übermittelt, aufgrund dessen die 3D-Objekte in den entsprechenden Tabellenzeile *n* und *n+1* der Tabelle „ferry queue“ getauscht werden.²⁷

²⁷ Zum Vergleich, wie ein Tausch auf der Benutzeroberfläche nachvollzogen und auf seine Plausibilität hin geprüft und an die 3D-Szene übermittelt bzw. ggfalls. verweigert wird, vgl. Quellcode der Dateien `tab.php` (Online-Version) bzw. `tab.html` (Offline-Version) auf der CD-ROM.

2.2.5 Die Webseite als Benutzeroberfläche

Der Browser fungiert als Benutzeroberfläche der Interaktiven 3D-Visualisierung. Dies ist zum einen ein Vorteil, weil sie auf der weit verbreiteten Windows-Plattform von den meisten Benutzern sofort gestartet werden kann. Zum anderen ergeben sich daraus alle nachteiligen Konsequenzen, die HTML mit sich bringt, wie z. B. mangelnde Formatierungsmöglichkeiten der Formularfelder. Aus Kapazitätsgründen wurde auf die Verwendung von gefälligen Flash-Elementen verzichtet. Mittels der Skriptsprache PHP, die auf dem Web-Server des IPA verwendet wird, ist aber zumindest die dynamische Anzeige der Formularfelder möglich, d. h. die Anzahl der Längen-Auswahlfelder und der Auswahlfelder im Eingabebereich für die Auftragsreihenfolge entspricht immer der eingestellten Brammenzahl.²⁸

Anmerkung: Die Offline-Version auf der CD-ROM enthält keinen PHP-Code. Daher sind exemplarisch 5 Brammen auf Strang 1 und 3 Brammen auf Strang 2 eingestellt.

Unzulässige Eingaben werden abgefangen. Im Sinne einer möglichst einfachen Programmierung der Plausibilitätsprüfungen bei der Auftragsreihenfolgenänderung kann der Benutzer an einer Reihenfolgenposition immer nur die betreffende Bramme mit ihrer Nachfolgebramme tauschen. Ein Tausch ist nicht zulässig, wenn die Nachfolgebramme vom gleichen Strang ist, da diese ja die Vorgängerbramme überholen müsste.

Der Besucher der IPA-Webseite wird von der Projektbeispielseite „Simulations- und Visualisierungsplattform SMS Demag AG“ der Abteilung Technische Produktionsplanung im Bereich Automatisierung über einen Link auf eine Willkommenseite geführt, die dem Besucher die Interaktive 3D-Visualisierung vorstellt und ihm aufzeigt, welche Bedienmöglichkeiten es gibt. Die nachfolgende Seite schließlich enthält die Interaktive 3D-Visualisierung.

²⁸ vgl. PHP-Code der Datei tab.php im Verzeichnis online auf der CD-ROM

Die Produktion

Eine Schritt-für-Schritt-Anleitung

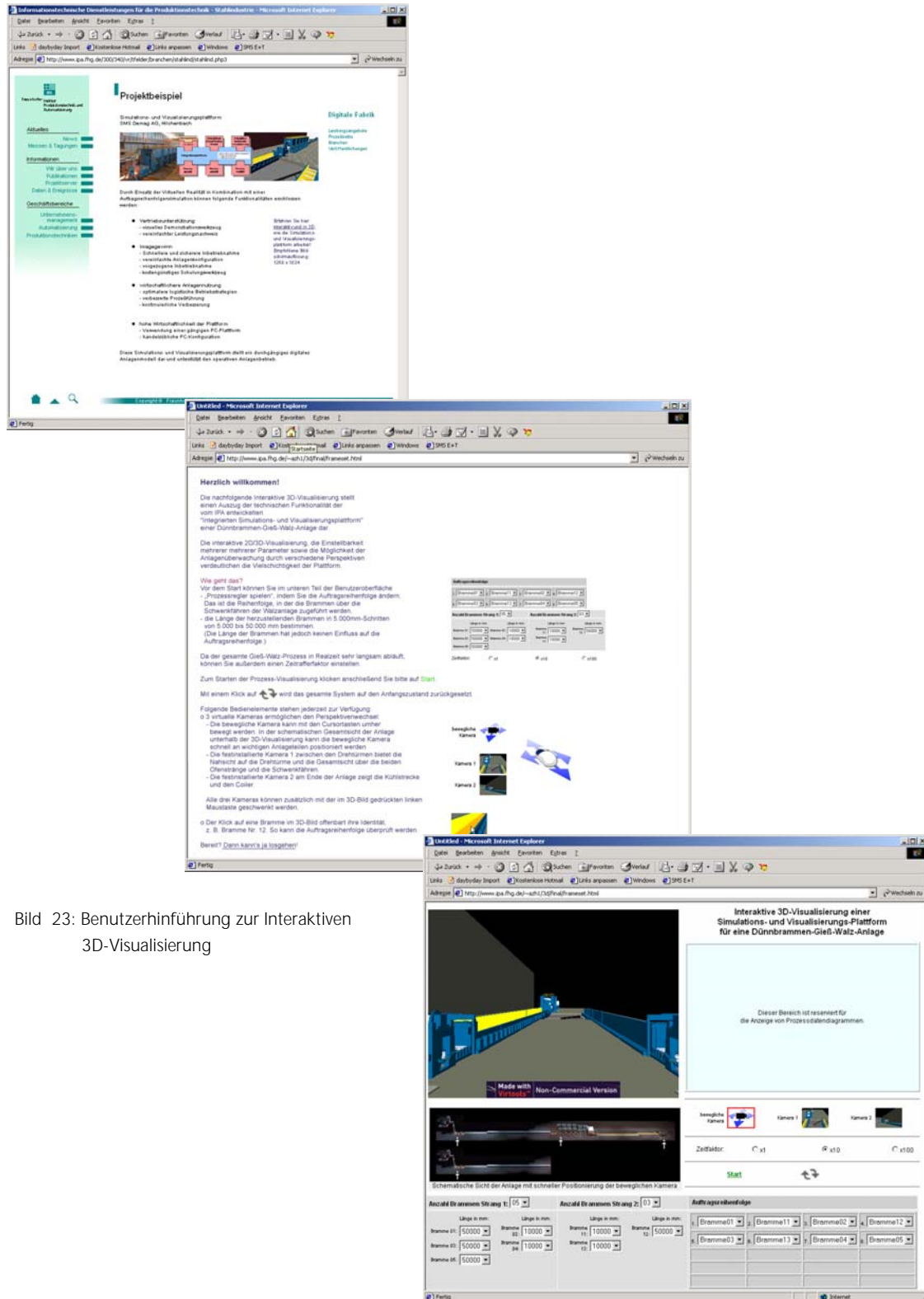


Bild 23: Benutzerhinführung zur Interaktiven 3D-Visualisierung

Die Produktion

Eine Schritt-für-Schritt-Anleitung

Ergänzend muss noch festgehalten werden, dass das *NeMo*-PlugIn, der *Web Player* sich leider auf einigen Systemen nicht immer fehlerfrei installieren lässt. Manchmal stürzt der Browser bei der Installation des PlugIns ab, beim nächsten Start des Browsers funktioniert das PlugIn jedoch. Dieses Verhalten wurde quantitativ gleichermaßen sowohl beim Internet Explorer 4.x und 5.x als auch beim Netscape Navigator in der Version 4.x beobachtet.

2.3 Weitere verwendete Softwares

Für die Realisierung der interaktiven 3D-Visualisierung wurden neben dem 3D-Autorenwerkzeug folgende Produkte eingesetzt:

- Kinetix 3D Studio Max 3.1 (Windows)
- Adobe Photoshop 6 (Windows)
- Silicon Graphics CosmoWorlds 1.0.3 (Unix)
- Medit 2.1p (Unix)
- Allaire HomeSite 4.5a

Kinetix 3D Studio Max 3.1

Die weit verbreitete 3D-Modellierungs- und Animations-Software wurde eingesetzt, um folgende Arbeitsschritte durchzuführen:

2. Zusammensetzen der Anlagenteile zu einer Gesamtszene und exakte Positionierung der Anlagenteile
3. Materialienreduktion
4. Vorbereitung diverser Hilfsobjekte für Animationen und Kameras
5. Positionieren von Lichtern

Adobe Photoshop 6

Die gängige Bildbearbeitungs-Software kam zum Einsatz für die Bearbeitung der Texturen und das Erstellen der auf der Webseite verwendeten Grafiken.

Silicon Graphics CosmoWorlds 1.0.3

CosmoWorlds ist selbst ein VRML-basierendes 3D-Autoren-Werkzeug, das in diesem Projekt wegen seines überzeugenden Polygonreduzierungswerkzeugs verwendet wurde.

Die Produktion

Weitere verwendete Softwares

Medit 2.1p

Medit ist eine Modellierungs-Software und beherrscht unter anderem das LOD-Konzept. Medit ist das 3D-Format, in dem die Original-Anlagenteile vorliegen. Schlussbetrachtung

Allaire HomeSite 4.5a

Mit dieser Software zur Produktion von Webseiten wurde die HTML-Seite der Benutzeroberfläche sowie der dazugehörige JavaScript- und PHP-Code geschrieben.

3 Schlussbetrachtung

Zum Zeitpunkt der Fertigstellung dieser Arbeit ist die Interaktive 3D-Visualisierung noch nicht fertig gestellt: Es fehlt die exemplarische Anzeige von Prozessdaten-Diagrammen der Anlagenteile und die Benutzeroberfläche ist nicht unter Netscape getestet.

Alle anderen in dieser Arbeit beschriebenen Anforderungen und Funktionen sind realisiert worden.

Die Interaktive 3D-Visualisierung erfüllt somit die wesentlichen Anforderungen und kann sowohl auf der IPA-Webseite als auch offline (mit einer festen Anzahl von Brammen) eingesetzt werden.

Trotz einiger Unzulänglichkeiten von *NeMo*, das in der verwendeten Version noch nicht den Kinderschuhen entwachsen ist, macht es Spaß, damit zu arbeiten und die Ergebnisse zu sehen.

Das Konzept der grafischen Skripte hat sich aber für die Projektarbeit als unzulänglich erwiesen. Die schlechte Dokumentierbarkeit der Skripte und ihre eingeschränkte Wiederverwendbarkeit erschweren das explorative Entwickeln einer Anwendung.

Wünschenswert wäre zumindest die alternative Möglichkeit zur Benutzung von geschriebenem Code, ähnlich wie es z. B. bei Flash mittlerweile möglich ist. Die aktuelle Versionsnummer 5 von Flash zeigt aber auch, dass dieses hervorragende 2D-Animationswerkzeug einen gewissen Reifeprozess hatte. Mit einer entsprechenden Marktbearbeitung und qualitativen Verbesserungen könnte *NeMo* bzw. *Virtools* ein *de facto*-Standardformat für interaktive 3D-Inhalte online und offline werden. Die rege Frequentierung des Benutzerforums von Virtools zeigt, dass das Produkt grundsätzlich gut aufgenommen wird, gleichzeitig haben aber vor allem Anfänger - zu denen der Autor sich auch noch zählt - enorme Anlaufschwierigkeiten. In diesem Projekt führten diese Schwierigkeiten zu etlichen Verzögerungen.

Die erzielten Ergebnisse veranlassten im IPA Überlegungen, kommende Weiterentwicklungen von *Virtools* zur Visualisierung in kleinerem Maßstab in ähnlichen Plattformen wie der SVP einzusetzen.

Vielleicht noch mehr als bei der „klassischen“ 3D-Animationsarbeit gilt für „Interaktives 3D“-Projekte, dass eine äußerst präzise Planung und Vorausschau absolut notwendig ist, um alle Eventualitäten zu erkennen und einen reibungslosen Projektverlauf zu gewährleisten.

4 Literatur- und Quellenverzeichnis

Hase, Hans-Lothar: Dynamische virtuelle Welten mit VRML 2.0, dpunkt-Verlag, 1. Auflage 1997

IPA: Dokumentation Simulationsplattform, Version 2.0 vom 25.6.2000

SGL: CosmoWorlds 1.0.3 Online-Dokumentation, 1998

www.cryonetworks.com Webseite des Software-Unternehmens Cryonetworks

www.eecs.tulane.edu/www/Terry/OpenGL/Introduction.html

OpenGL Survival Kit Tutorial, Nicole Deflaux Terry

www.glossar.de Glossar für Internet-, IT- und Bildverarbeitungsbegriffe

www.interest.de Glossar Telekommunikation

www.highend3d.com Eine Plattform für 3D-Künstler

www.ipa.fhg.de Webseite des Fraunhofer IPA

www.shout3d.com Webseite des Produkts Shout3D von Eyematic

www.sms-demag.de Webseite der SMS Demag AG

www.spazz3d.com Webseite des Produkts Spazz3D von Virtock

www.theswapmeet.com Eine Plattform für Virtools-Anwender mit einem gut frequentierten Anwenderforum, das vom Support-Team von Virtools gepflegt wird

www.uni-giessen.de/~gm01/lexikon/bildformate.shtml

Abkürzungsverzeichnis gängiger Bildformate

www.virtools.com Webseite des Software-Unternehmens Virtools und des gleichnamigen 3D-Autoren-Werkzeugs

www.web3d.org Webseite des Web3D Consortiums

5 Anlagen

Anlage A: Alphabetisch sortierte Liste aller *NeMo Building Blocks*

Behavior Name	Category
2d Picking	Narratives
3D Wave Player	Sounds/3D
Activate Channel	Materials-Textures/Channel
Activate Link	3D Transformations/Nodal Path
Activate Node	3D Transformations/Nodal Path
Activate Object	Narratives
Activate Script	Narratives
Activate State	Narratives
Add Channel	Materials-Textures/Channel
Add Child	3D Transformations/Basic
Add Elements	Logics/Array
Add Mesh	Mesh Modifications/Multi Mesh
Add Object To Group	Logics/Groups
Add Obstacle	Collisions/Obstacle
Add Scale	3D Transformations/Basic
Add To Group	Logics/Groups
All But One	Logics/Streaming
Animation Synchronizer	Characters/Animation
Avoiding obstacles	Collisions/Influence
Bend	Mesh Modifications/Deformation
Bezier Interpolator	Logics/Interpolator
Bezier Progression	Logics/Loops
Binary Memory	Logics/Streaming
Binary Switch	Logics/Streaming
BitmapText Display	Visuals/Text
Blink	Visuals/FX
Bool Event	Logics/Streaming
Bounding Sphere Intersection	Collisions/Test
Box Box Intersection	Collisions/Test
Box Face Intersection	Collisions/Test
Broadcast Message	Logics/Message
Calculator	Logics/Calculator
Camera Color Filter	Cameras/FX
Camera Orbit	Cameras/Movement
CD Player	Sounds
Change Texture Slot	Materials-Textures/Basic
Change Value If	Logics/Array
Character Controller	Characters/Movement
Character Go To	Characters/Movement
Character Keep On Floor	Characters/Constraint

Anlage A: Alphabetisch sortierte Liste aller *NeMo Building Blocks* (Fortsetzung)

Behavior Name	Category
Character Keep On Floor Limits	Characters/Constraint
Character Keep On Floor V2	Characters/Constraint
Character Path Follow	Characters/Movement
Character Path Follow 2	Characters/Movement
Character Prevent From Collision	Collisions/Character
Check For Message	Logics/Message
Clear	Logics/Array
Clouds Around	World Environments/Background
Collision Detection	Collisions/3D Entity
Column Product	Logics/Array
Column Sum	Logics/Array
Column Transform	Logics/Array
Columns Operate	Logics/Array
Counter	Logics/Loops
Create Group From Array	Logics/Array
Create Merged Animation	Characters/Animation
Create Merged Animation	Characters/Animation
Create Nodal Path	3D Transformations/Nodal Path
Cross Waves	Sounds
Deactivate Object	Narratives
Deactivate Script	Narratives
Deactivate State	Narratives
Declare Floors	Collisions/Floors
Declare Obstacles	Collisions/Obstacle
Display Omni Light	Lights/FX
Display Progression Bar	Visuals/2D
Display Score	Visuals/Text
Dolly	Cameras/Basic
Edit 2D Entity	Visuals/2D
Enhanced Character Controller	Characters/Movement
Enter Critical Section	Logics/Synchro
Env. Mapping	Visuals/FX
Exclude From Animation	Characters/Animation
Explode	Mesh Modifications/Deformation
Face Face Intersection	Collisions/Test
FIFO	Logics/Streaming
Fill Layer	Grids
Find Curved Path	3D Transformations/Nodal Path
Flash Color	Lights/FX
Floor Manager Setup	Collisions/Floors

Anlage A: Alphabetisch sortierte Liste aller *NeMo Building Blocks* (Fortsetzung)

Behavior Name	Category
Fps	Visuals/Text
Frequency Control	Sounds/Control
Get Component	Logics/Calculator
Get Current Camera	Cameras/Montage
Get Current Scene	Narratives
Get Data Message	Logics/Message
Get Delta Time	Logics/Calculator
Get Elements	Logics/Array
Get Highest	Logics/Array
Get Lowest	Logics/Array
Get Mouse Position	Controllers/Mouse
Get Mouse Relative Position	Controllers/Mouse
Get Nearest	Logics/Array
Get Nearest Floors	Collisions/Floors
Get Nearest In Group	Logics/Groups
Get Nearest Object	Characters/Basic
Get Pos From Value	Grids
Get Proportional Screen Pos	Logics/Calculator
Get Row	Logics/Array
Get Screen Proportional Pos	Logics/Calculator
Get Square From 3dpos	Grids
Go To Node	3D Transformations/Nodal Path
Group Iterator	Logics/Groups
Has Attribute	Logics/Attribute
Hide	Visuals/Show-Hide
Hide 2D Entity	Visuals/Show-Hide
Homing Missile	Collisions/Influence
HSV Color Interpolator	Logics/Interpolator
IBCQ	Logics/Loops
Identity	Logics/Calculator
IK Position	Characters/IK
Interpolator Color	Logics/Interpolator
Interpolator Float	Logics/Interpolator
Interpolator Int	Logics/Interpolator
Interpolator Matrix	Logics/Interpolator
Interpolator Orientation	Logics/Interpolator
Interpolator Vector	Logics/Interpolator
Is In Group	Logics/Groups
Is In View Frustum	Logics/Test
Is State Active	Narratives

Anlage A: Alphabetisch sortierte Liste aller *NeMo Building Blocks* (Fortsetzung)

Behavior Name	Category
Iterator	Logics/Array
Iterator If	Logics/Array
Joystick Controller	Controllers/Joystick
Joystick Mapper	Controllers/Joystick
Joystick Waiter	Controllers/Joystick
Keep At Constant Distance	3D Transformations/Constraint
Key Event	Controllers/Keyboard
Key Waiter	Controllers/Keyboard
Keyboard Controller	Controllers/Keyboard
Keyboard Mapper	Controllers/Keyboard
Launch Scene	Narratives
Layer Slider	Grids
Leave Critical Section	Logics/Synchro
Level Of Detail	Mesh Modifications/Multi Mesh
LIFO	Logics/Streaming
Linear Progression	Logics/Loops
Load	Logics/Array
Look At	3D Transformations/Constraint
Look At Pos	3D Transformations/Constraint
Make Transparent	Visuals/FX
Mesh Lighting	Visuals/FX
Mesh Morpher	Mesh Modifications/Deformation
Mesh Morpher	Mesh Modifications/Deformation
Mesh Texture Displace	Mesh Modifications/Deformation
Midi Player	Sounds
Mimic	3D Transformations/Constraint
Mini Calculator	Logics/Calculator
Motion Blur	Visuals/FX
Mouse Waiter	Controllers/Mouse
Move To	3D Transformations/Movement
Movie Player	Materials-Textures/Animation
Noise	Mesh Modifications/Deformation
Object between	Logics/Test
Object Keep On Floor	3D Transformations/Constraint
Object Keep On Floor V2	3D Transformations/Constraint
Objects With Attribute Iterator	Logics/Attribute
One At A Time	Logics/Streaming
Op	Logics/Calculator
Orthographic Zoom	Cameras/Basic
Panning Control	Sounds/Control

Anlage A: Alphabetisch sortierte Liste aller *NeMo Building Blocks* (Fortsetzung)

Behavior Name	Category
Parameter Selector	Logics/Streaming
Particle Systems	Particles
PathFollow	3D Transformations/Path
Per Second	Logics/Calculator
Planar Filter	Visuals/FX
Play Animation 3D Entity	3D Transformations/Animation
Play Global Animation	3D Transformations/Animation
Position On Curve	3D Transformations/Path
Prevent From Collision	Collisions/3D Entity
Priority	Logics/Streaming
Proximity	Logics/Test
Random	Logics/Calculator
Random Switch	Logics/Streaming
Ray Intersection	Logics/Test
Remove Attribute	Logics/Attribute
Remove Channel	Materials-Textures/Channel
Remove Elements	Logics/Array
Remove From Group	Logics/Groups
Remove If	Logics/Array
Remove Mesh	Mesh Modifications/Multi Mesh
Remove Object From Group	Logics/Groups
Remove Row	Logics/Array
Rendez Vous	Logics/Synchro
RestoreIC	Narratives
Reverse	Logics/Array
Rotate	3D Transformations/Basic
Rotate Around	3D Transformations/Basic
Row Search	Logics/Array
Row Test	Logics/Array
Select Mesh	Mesh Modifications/Multi Mesh
Send Message	Logics/Message
Send Message To Group	Logics/Message
Sequencer	Logics/Streaming
Set 3DSprite Mode	Visuals/Sprite
Set Ambient	Materials-Textures/Basic
Set Animation Frame	Characters/Animation
Set Animation Step	Characters/Animation
Set Animation Step on 3D entity	3D Transformations/Animation
Set Attribute	Logics/Attribute
Set Background Color	World Environments/Background

Anlage A: Alphabetisch sortierte Liste aller *NeMo Building Blocks* (Fortsetzung)

Behavior Name	Category
Set Background Image	World Environments/Background
Set Background Material	World Environments/Background
Set Bodypart Animation Frame	Characters/Animation
Set Camera Target	Cameras/Basic
Set Clipping	Cameras/Basic
Set Color Key	Materials-Textures/Basic
Set Color Key 2d	Visuals/2D
Set Component	Logics/Calculator
Set Cone	Sounds/3D
Set Constant Attenuation	Lights/Basic
Set Dest Blend	Materials-Textures/Basic
Set Diffuse	Materials-Textures/Basic
Set Elements	Logics/Array
Set Emissive	Materials-Textures/Basic
Set Euler Orientation	3D Transformations/Basic
Set Fall Off	Lights/Basic
Set Fill Mode	Materials-Textures/Basic
Set Floor Reference Object	Characters/Basic
Set Fog	World Environments/Background
Set FOV	Cameras/Basic
Set Grid Priority	Grids
Set Light Color	Lights/Basic
Set Light Range	Lights/Basic
Set Light Target	Lights/Basic
Set Light Type	Lights/Basic
Set Linear Attenuation	Lights/Basic
Set Lit Mode	Materials-Textures/Basic
Set Local Matrix	3D Transformations/Basic
Set Material	Materials-Textures/Basic
Set Material 3DSprite	Materials-Textures/Basic
Set Orientation	3D Transformations/Basic
Set Parent	3D Transformations/Basic
Set Position	3D Transformations/Basic
Set Power	Materials-Textures/Basic
Set Prelit Color	Materials-Textures/Basic
Set Projection	Cameras/Basic
Set Quadratic Attenuation	Lights/Basic
Set Range	Sounds/3D
Set Row	Logics/Array
Set Scale	3D Transformations/Basic

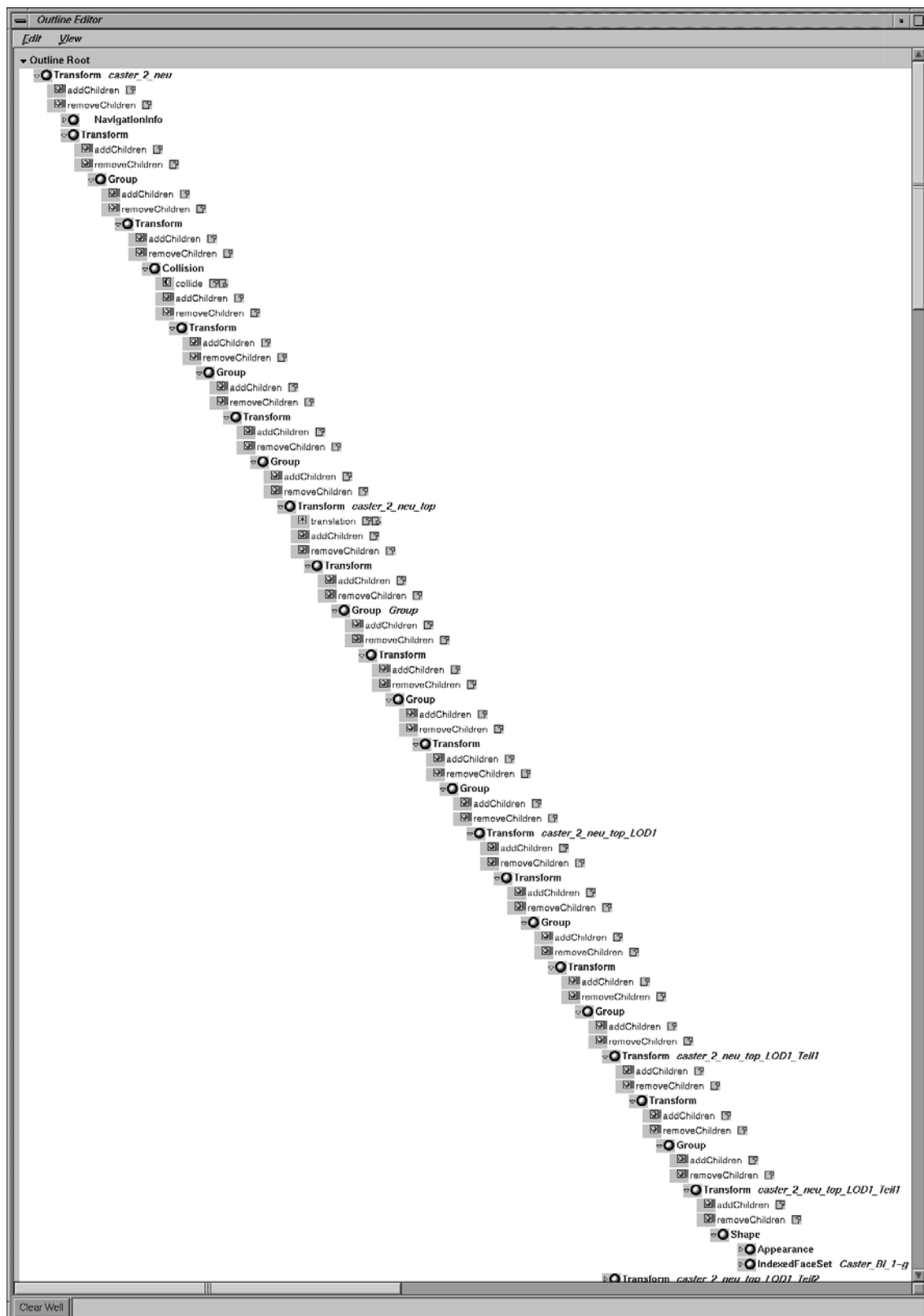
Anlage A: Alphabetisch sortierte Liste aller *NeMo Building Blocks* (Fortsetzung)

Behavior Name	Category
Set Shade Mode	Materials-Textures/Basic
Set Specular	Materials-Textures/Basic
Set Specular Flag	Lights/Basic
Set Square From 3dpos	Grids
Set Src Blend	Materials-Textures/Basic
Set Texture	Materials-Textures/Basic
Set Texture Mag	Materials-Textures/Basic
Set Texture Min	Materials-Textures/Basic
Set Transparent	Materials-Textures/Basic
Set Two Sided	Materials-Textures/Basic
Set World Matrix	3D Transformations/Basic
Set Wrap Mode	Materials-Textures/Basic
Set ZBuf	Visuals/FX
Set Zoom	Cameras/Basic
Set ZOrder	Visuals/FX
SetAsActiveCamera	Cameras/Montage
SetChannelDestBlend	Materials-Textures/Channel
SetChannelMaterial	Materials-Textures/Channel
SetChannelSrcBlend	Materials-Textures/Channel
Show	Visuals/Show-Hide
Show 2D Entity	Visuals/Show-Hide
Show Bounding Box	Visuals/Show-Hide
Show Mouse Cursor	Visuals/Show-Hide
Shuffle	Logics/Array
Simple Shadow	Visuals/FX
Sine Deform	Mesh Modifications/Deformation
Sky Around	World Environments/Background
Solid Trail	Visuals/FX
Sort	Logics/Array
Sound Manager Setup	Sounds/3D
Specific Bool Event	Logics/Streaming
Sprite Movie Player	Visuals/Sprite
Sprite Multi Angle	Visuals/Sprite
Stop & Go	Logics/Streaming
Stretch	Mesh Modifications/Deformation
Swap Rows	Logics/Array
Switch If Square	Grids
Switch On Key	Controllers/Keyboard
Switch On Message	Logics/Message
Switch On Parameter	Logics/Streaming

Anlage A: Alphabetisch sortierte Liste aller *NeMo Building Blocks* (Fortsetzung)

Behavior Name	Category
Taper	Mesh Modifications/Deformation
Test	Logics/Test
Test Bounding Box Y Overlapping	Collisions/3D Entity
Text Display	Visuals/Text
Texture Render	Visuals/FX
Texture Scroller	Materials-Textures/Animation
Texture Sine	Materials-Textures/Animation
TextureProjector	Visuals/FX
Threshold	Logics/Calculator
Time Bezier Interpolator	Logics/Interpolator
Timer	Logics/Loops
Translate	3D Transformations/Basic
Twist	Mesh Modifications/Deformation
Unique	Logics/Array
Unlimited Controller	Characters/Movement
Update Nodal Path	3D Transformations/Nodal Path
Uses Z information	Visuals/FX
Value Count	Logics/Array
Variation	Logics/Calculator
Vertices Translate	Mesh Modifications/Local Deformation
Vertigo	Cameras/FX
Volume Control	Sounds/Control
Wait For All	Logics/Synchro
Wait Message	Logics/Message
Wave Player	Sounds
While	Logics/Loops
Write	Logics/Array

Anlage B: Screenshot des Outline-Editors von CosmoWorlds, von der Knotenwurzel bis zur ersten auftretenden Geometrie im Modell des Casters



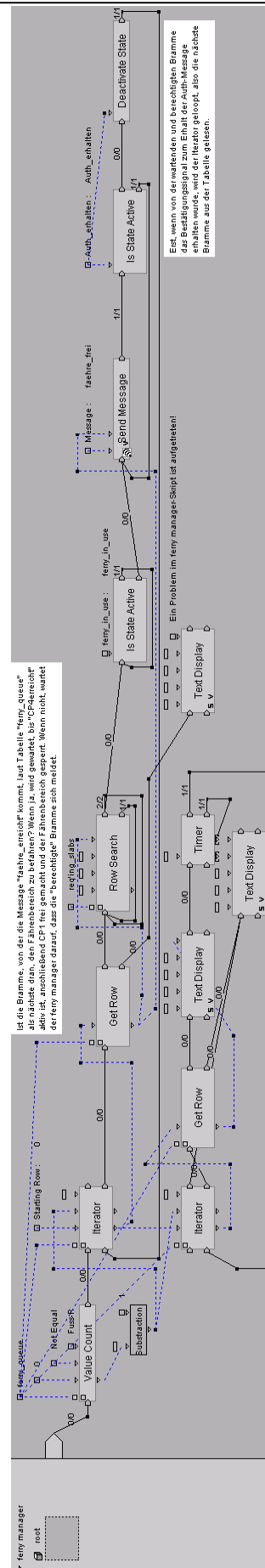
Anlage C: Tabelle THREEED-Events der Integrierten Simulations- und Visualisierungs-Plattform (Quelle: Dokumentation Simulationsplattform, IPA)

Event bzw. Kennung	Eventbeschreibung
Caster1 (Section1) und	Caster2 (Section2)
Event 0	Simulationsstart
Event 1	Drehung der Pfanne zum Giessen zum Zeitpunkt Model_Time in Section_ID. In der Spalte SLAB_speed wird die Dauer [in Sekunden] hineingeschrieben.
Event 2	Drehung der Pfanne weg vom Giessen zum Zeitpunkt Model_Time in Section_ID In der Spalte SLAB_speed wird die Dauer [in Sekunden] hineingeschrieben.
Event 3	Füllbeginn des Tundish zum Zeitpunkt Model_Time in Section_ID. Nach Verstreichen der Dauer SLAB_speed [in Sekunden] wird mit Event 1 der eigentliche Gießbeginn eingeleitet.
Event 4	Füllende des Tundish zum Zeitpunkt Model_Time in Section_ID. Nach Verstreichen der Dauer SLAB_speed [in Sekunden] wird mit Event 2 das eigentliche Gießende eingeleitet.
Event 10	Gießbeginn in Sektion Section_ID mit Gießgeschwindigkeit SLAB_speed. SLAB_ID und POSITION sind hier immer 0.
Event 20	Gießende in Sektion Section_ID mit Gießgeschwindigkeit SLAB_speed. SLAB_ID und POSITION sind hier immer 0.
Event 30	Simulationsende
Event 40	Übertragung des neuen Zeitraffer-Faktors Das Feld SECTION_ID enthält dabei den Faktor (1-10). SLAB_ID, POSITION und SLAB_SPEED werden zu 0.
Event 121 bzw.221	Austrittszeit des Brammenkopfes aus dem (Caster1 bzw. Caster2) = Austritt aus der Bramme aus dem Caster. In Tabelle THREEED_EVENTS wird die Brammenlänge im Feld POSITION eingetragen.
Event 50	Befehl zum Öffnen des Ofeneintritts zum Zeitpunkt Model_Time in Section_ID.
Event 60	Befehl zum Schliessen des Ofeneintritts zum Zeitpunkt Model_Time in Section_ID.
Ofen1 (Section3) und	Ofen2 (Section4)
Event 304 bzw. 404	Enthält Offsetwert im Ofen1 (Section_ID = 3) bzw. Ofen2 (Section_ID = 4) durch die Position_ID Sollte mindestens 1ms nach Event 121 bzw. 221 in die Datenbank geschrieben werden. Da das Event 304 bzw. 404 erst generiert wird, nachdem die Bramme geschnitten ist, stellt diese Anforderung momentan kein Problem dar.
Fähre2 (Section5) und	Fähre1 (Section6)
Event 500	Eintritt der Bramme in den Fahrenbereich
Event 502	Fährenschwenkvorgang wird gestartet (Bramme befindet sich dabei in Section_ID = 5) Die hier übergebene Brammengeschwindigkeit entspricht der (konstanten) Transfergeschwindigkeit der Bramme beim Wechsel der Linien.
Event 504	Enthält Offsetwert im Fahrenbereich des Strang2 (Section_ID = 5) der in der Position_ID übergeben wird
Event 603	Fährenschwenkvorgang ist beendet, d.h. Fähre befindet sich wieder in Parallel-Stellung und

Anlagen

Event bzw. Kennung	Eventbeschreibung
	Bramme fährt weiter.
Event 604	Enthält Offsetwert im Fahrenbereich des Strang1 (Section_ID = 6) in Form der Position_ID
Event 605	Austritt der Bramme aus dem Ofen. Zu diesem Zeitpunkt kann VR4KIN die Preset-Daten aus THREED_PRESETS lesen. Die mit diesem Event übertragene Brammengeschwindigkeit entspricht der Einlaufgeschwindigkeit in das erste Walzgerüst.
Event 606	Befehl zum Öffnen des Ofenendes zum Zeitpunkt Model_Time.
Event 607	Befehl zum Schliessen des Ofenendes zum Zeitpunkt Model_Time.

Anlage E: NeMo-Skript „ferry manager“



Anlage F: CD-ROM

Die beiliegende CD-ROM enthält den Stand der Interaktiven 3D-Visualisierung zum Zeitpunkt der Abgabe dieser Arbeit.

Die finale Version wird auf der Webseite des IPA eingesetzt werden und mit bis zu insgesamt 20 Brammen auch die exemplarischen Prozessdatendiagramme anzeigen können

Gestartet wird die offline-Version der Interaktiven 3D-Visualisierung mit der Datei **willkommen.html** im offline-Verzeichnis. Da sie nur mit Internet Explorer getestet wurde, wird von Verwendung des Netscape Navigators abgeraten.

Für die vollständige Anzeige ist eine Bildschirmauflösung von 1280 x 1024 Punkten erforderlich.

Beim ersten Aufruf der Interaktiven 3D-Visualisierung ist eine Online-Verbindung erforderlich, damit die automatische Installation des PlugIns erfolgen kann.

Desweiteren beinhaltet die CD-ROM

- die Diplomarbeit im Microsoft Word 2000- und im PDF-Format
- die NeMo-Projektdatei der Diplomarbeit
- NeMo Evaluation (nicht zur kommerziellen Verwendung)